24

Generalized "Yoking-Proofs" and Inter-Tag Communication

Leonid Bolotnyy and Gabriel Robins
Department of Computer Science, University of Virginia, USA

1. Introduction

Some radio-frequency identification (RFID) scenarios require a proof of action (e.g., that a group of objects tagged with RFID tags were identified simultaneously). For example, pharmaceutical distributors may want to prove that a bottle of medicine was sold together with its instructions leaflet [Juels, 2004]; manufacturers may want to prove that safety devices were sold together with a tool or that a number of matching parts were delivered simultaneously; banking centers or security stations may want to prove that several forms of ID were read simultaneously; meeting organizers may want to prove that a group of people were present together at a meeting, etc. Third-parties can verify the validity of such proofs. For examples above, the verifying third parties can be regulatory agencies, company headquarters, etc.

We seek to ensure that if a group of tags is not read nearly-simultaneously, an entity (RFID reader) will not be able to forge a proof that they were by constructing a valid forged proof. Inspired by this problem, Ari Juels developed a protocol that creates such a proof for a *pair* of RFID tags [Juels, 2004]. He left open for future research the problem of generalizing his protocol to three or more tags. We develop a methodology that generalizes yoking-proof protocols to arbitrarily large groups of tags. We also define an *anonymous yoking* problem and propose an efficient solution for it [Bolotnyy & Robins, 2006]. Finally, we show how these yoking protocols can be sped up.

2. Assumptions

We assume that RFID tags are passive and have limited computational capabilities. We require that tags be able to execute keyed hash functions and store state information such as a key, a counter, and some data computed during the protocol execution. These requirements can be satisfied in practice by Class-2 Generation-2 EPC tags [EPCglobal, 2006]. Since we assume that tags are passive, the tags cannot communicate directly with each other, but they can communicate with each other indirectly through the reader. For now, we also assume that an adversary cannot physically steal tags' secret information. Later we discuss how this requirement can be relaxed.

Our verifier is assumed to be a trusted and computationally powerful machine. The verifier is considered to be off-line in the sense that it does not have to verify the proof immediately after it is created, and it does not need to communicate with the tags during the proof

Source: Development and Implementation of RFID Technology, Book edited by: Cristina TURCU,
ISBN 978-3-902613-54-7, pp. 554, February 2009, I-Tech, Vienna, Austria

construction. A reader communicating with the tags is assumed to be adversarial, and we want the protocol to be secure against a reader that attempts to create the yoking proof without reading all the tags within the required time bounds. The tags are assumed to be not impaired by an adversary, and tags do not collude with an adversary.

Replays of previously constructed valid proofs (replay attacks) may or may not be considered a threat, depending on the application. In our generalized yoking-proof protocol, we consider adversarial replay attacks to be a viable threat, and thus we design the protocol accordingly. To avoid replay attacks, the proof verifier stores some information about previous correct proofs. The verifier is not required to store this information if replays of valid proofs are not considered to be attacks. This will be elaborated upon in the discussion following the protocol specification.

We require that the tag accessed first by the reader be able to implement a timeout after a specific time period t has elapsed. Perhaps surprisingly, timeouts can be implemented on clock-less RFID tags. FCC regulations require the reader to change the communication frequency of a tag-reading protocol within 400ms. Changing the communication frequency in the middle of the protocol execution will likely result in loss of power on-board a tag and cause protocol termination. Therefore, the FCC regulation can serve as the clock for honest (law-obedient) readers. However, if the reader is malicious and violates these FCC regulations, a capacitor discharge rate on-board a tag can be used for protocol timing [Juels, 2004].

3. Basic protocol for a pair of tags

We now briefly describe Ari Juels' "yoking"[1] protocol for a pair of tags [Juels, 2004]. Assume that an RFID system contains n tags, which are denoted by $T_1, T_2, \ldots, T_n$. Each tag T_i is assigned a unique key x_i and a counter c_i both are d bits long. A tag has the ability to compute a keyed hash function and a standard message authentication code, which is likely to be implemented as a keyed hash function, such as HMAC, in order to simplify the circuit. A reader will read two tags and produce a proof P that both tags were read near-simultaneously, i.e. within t time units. The verifier V knows all key assignments to tags and will verify that the proof P is valid (not forged).

Let $f : \{0, 1\}^d \times \{0, 1\}^* \rightarrow \{0, 1\}^d$ be a keyed hash function, and let $MAC : \{0, 1\}^d \times \{0, 1\}^* \rightarrow \{0, 1\}^d$ denote a standard *message authentication code*. Let $f_x[m]$ and $MAC_x[m]$ denote computation of f and MAC respectively with a secret key x on input message m. The proof that tags T_A and T_B were scanned simultaneously is $P_{AB} = (A, B, c_A, c_B, m_{AB})$ where m_{AB} is defined in the protocol in Figure 1. To verify proof P_{AB}, the verifier computes $a' = (A, c_A, f_{x_A}[c_A])$, and $b' = (B, c_B, MAC_{x_B}[a', c_B])$, as well as $m'_{AB} = MAC_{x_A}[c_A, b']$, and then checks if $m_{AB} = m'_{AB}$. If t time units elapse, the first tag terminates the protocol, thus depriving the reader of the ability to construct a valid proof. Juels's protocol, as shown in Figure 1, has a minor, yet critical omission. Specifically, the counter value on the first tag is not incremented on timeout, allowing an adversary to violate the near-simultaneous object read requirement. Our group yoking protocol corrects this problem.

Juels also introduced a "Minimalist MAC", which may be implemented on lower-cost tags without encryption or hash function support. However, the protocol that he suggests for a

[1] The term "yoking" suggests the *joining together*, or the simultaneous presence of all the tags.

"yoking-proof" using a "Minimalist MAC" can only construct the proof once, which seems to diminish the original purpose of the protocol. If the reader is trusted, there is no need for such a protocol; on the other hand, if the reader is potentially untrusted and aborts the protocol after the first read, there will be no proof and no chance of ever re-creating a provably unforgeable proof in the future. Next, we discuss several works that unsuccessfully attempt to solve the generalized "yoking-proof" problem.

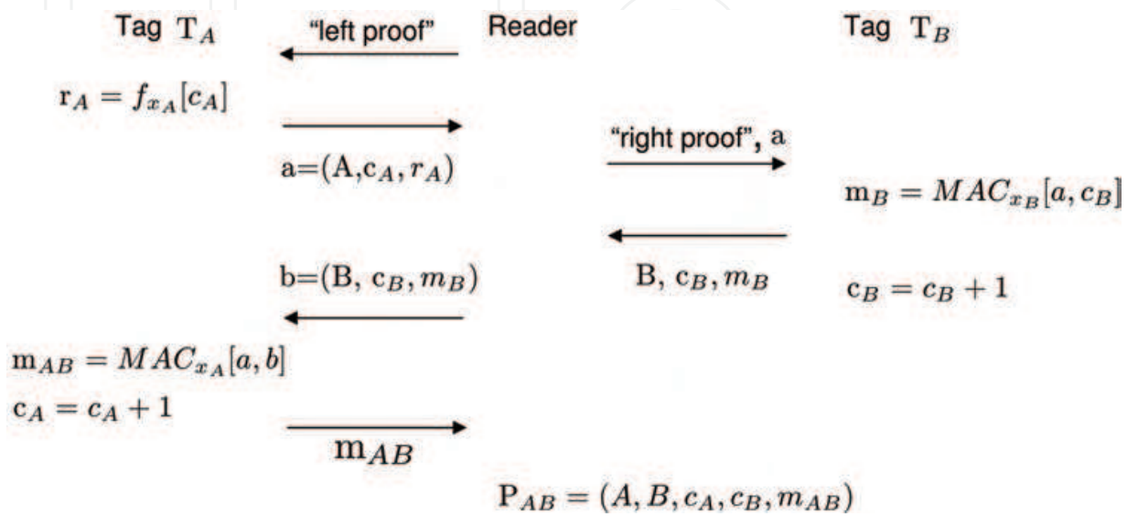

Fig. 1. A "yoking-proof" construction for a pair of RFID tags. The reader first communicates with tag T_A, then with tag T_B, and then with tag T_A again.

4. Related work

There have been several attempts to generalize the yoking-proof protocol. However, none of them solve the problem satisfactorily. Saito and Sakurai [Saito and Sakurai, 2005] apparently misunderstood Ari Juels' protocol for "strong MAC" and its modification for a "minimalist MAC". The authors presumed that in a minimalist "yoking-proof" each tag generates a random number for each proof, on which a keyed-MAC function is later applied, whereas Juels states that each tag is initialized with a one-time random number. Thus, the replay-attack the authors consider against the "minimalist-MAC" protocol is not applicable, since the "minimalist-MAC" protocol was designed for one-time use.

Saito and Sakurai suggest a group protocol which relies on time stamps provided by the back-end database [Saito & Sakurai, 2005]. In their scheme a reader receives a time stamp TS from the database, and it sends this timestamp to all the tags participating in the protocol. Each tag T_i computes $m_i = MAC_{x_i}[TS]$ and sends m_i back to the reader. In addition, the authors assume the existence of one powerful/leader tag among tags participating in the protocol. The reader sends all m_i to this leader tag, and the leader tag encrypts them together with the time stamp TS using encryption function SK keyed with a secret key x. Then, the leader tag sends the encryption result C_p to the reader, and the yoking proof is $P_n = (TS, C_p)$. Figure 2 shows their grouping protocol.

We discovered several flaws in Saito and Sakurai's solution. First, the assumption that one of the tags is more powerful than the others is not true in many practical scenarios. The second and main weakness of their protocol is that an untrusted reader can pick the time

stamp *TS* as a future time stamp, and then use it on one tag and much later on another, thus violating the near-simultaneous read requirement. Even if the time stamp *TS* is encrypted, the reader can still separate tag accesses in time, since each tag access is independent of the others.

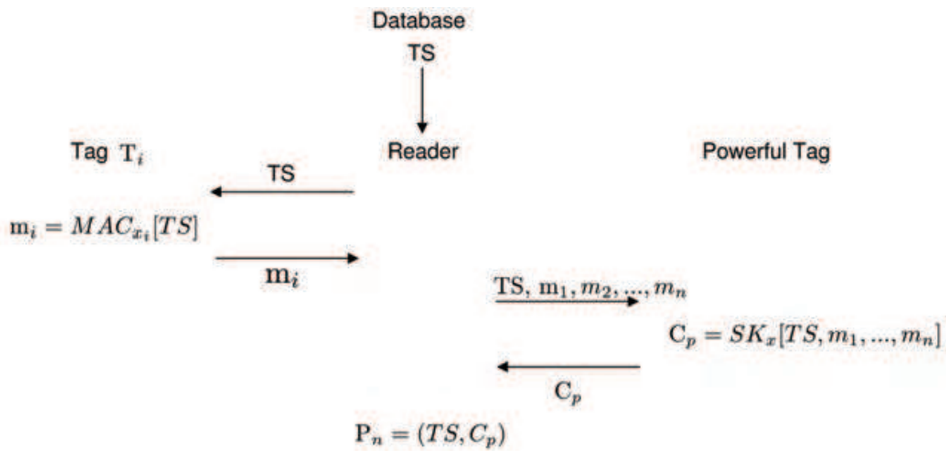

Fig. 2. Saito and Sakurai's protocol for a group of RFID tags.

Recently another paper [Piramuthu, 2006] independently observed the same problem with the "yoking-proofs" protocol of Saito and Sakurai [Saito & Sakurai, 2005]. Both papers [Saito & Sakurai, 2005] [Piramuthu, 2006] discuss a replay attack on Juels' one-time "yoking-proof". However, replay attacks are not an issue in Juels' Minimalist-MAC protocol [Juels, 2004] since the yoking-proof is a one-time proof. The one-time "yoking-proof" of Juels provides no security guarantees if the protocol is run more than once, and in fact, it is insecure in such re-run scenarios. The proposed "fix" of Piramuthu [Piramuthu, 2006] only works for a pair of tags, as opposed to an arbitrary number of tags. Moreover, the method of Piramuthu [Piramuthu, 2006] solves a different problem than the original problem formulation of Juels [Juels, 2004], since it relies on a random number the reader obtains from an on-line verifier, rather than accommodating an off-line verifier, as done in [Juels, 2004]. Our proposed solutions do not have these limitations.

Lastly, the work Peris-Lopez et al. [Peris-Lopez et al., 2007] repeats our observations of flaws in previous attempts to generalize yoking proofs. The authors [Peris-Lopez et al., 2007] offer their solution to our anonymous-yoking problem, discussed below, yet their solution does not satisfy the problem requirements. Specifically, it does not provide privacy since their protocol leaks the counter value on-board the tags. In addition, their solution is applicable only to 2 tags, requires an on-line verifier, and forces the reader to be trusted. It is worth mentioning that Peris-Lopez et al. [Peris-Lopez et al., 2007] state that our anonymous-yoking protocol takes $O(n^2)$, where n is the total number of tags in the system, to verify the yoking-proof for two tags. However, this bound is not tight since our anonymous-yoking protocol takes $\theta(k*n)$ where k is the number of tags participating in the yoking protocol and n is the total number of tags in the system. So, if the number of tags participating in the protocol is small, as is the case for most realistic scenarious, our protocol takes $\theta(n)$ time.

5. Our group yoking protocol

The idea of our generalized "yoking proof" for a group of tags is to construct a circular chain of mutually dependent MAC computations [Bolotnyy & Robins, 2006]. The purpose of

the construction is to ensure that if an untrusted reader "breaks the chain" (i.e. does not read all the tags within t time units), it will neither be able to mount a replay attack nor create a proof that will be accepted as valid by the verifier.

Using the same notation and definitions as above, let $x_1, x_2, \ldots, x_n$ be the secrets and let $c_1, c_2, \ldots, c_n$ be the counters stored on tags $T_1, T_2, \ldots, T_n$, respectively. Tag secrets are shared with the verifier. In our protocol below, we assume that the reader generates a proof for tags $T_1, T_2, \ldots, T_k$ ($k \leq n$) and queries them in that order. (Tags are queried based on their IDs or on the random numbers that they generate.) The first tag computes $r_1 = f_{x1}[c_1]$ and sends $a_1 = (1, c_1, r_1)$ to the reader. The reader will then send a_1 to the second tag. The second tag will compute $r_2 = MAC_{x2}[c_2, a_1]$ and send $a_2 = (2, c_2, r_2)$ to the reader. The reader will then send a_2 to the third tag, which will in turn perform the same computation as the second tag, i.e. $r_3 = MAC_{x3}[c_3, a_2]$ and send $a_3 = (3, c_3, r_3)$ to the reader. The reader will then send a_3 to the fourth tag and so on, until the last tag k has computed r_k and sent $a_k = (k, c_k, r_k)$ to the reader.

The reader then sends a_k to the first tag, which computes $m = MAC_{x1}[a_1, a_k]$ (assuming that t time units have not yet elapsed since the initial tag access), and sends m to the reader. The reader R creates a proof $P_{1,2,\ldots,k} = (1, 2, \ldots, k, c_1, c_2, \ldots, c_k, m)$. The protocol is shown in Figure 3. The numbers in ellipses in the figure indicate the communication order. To verify the proof $P_{1,2,\ldots,k}$, the verifier V performs the same computations as the tags 1, 2, . . . , k, maintaining the order, and compares the proof P that it generates to the proof $P_{1,2,\ldots,k}$ that the reader provided. If the proofs match, the verifier outputs *success*; otherwise, it outputs *failure*. The pseudocode of algorithms for tag initialization, the reader, a tag, and the verifier can be found in the author's Ph.D. thesis [Bolotnyy, 2007].

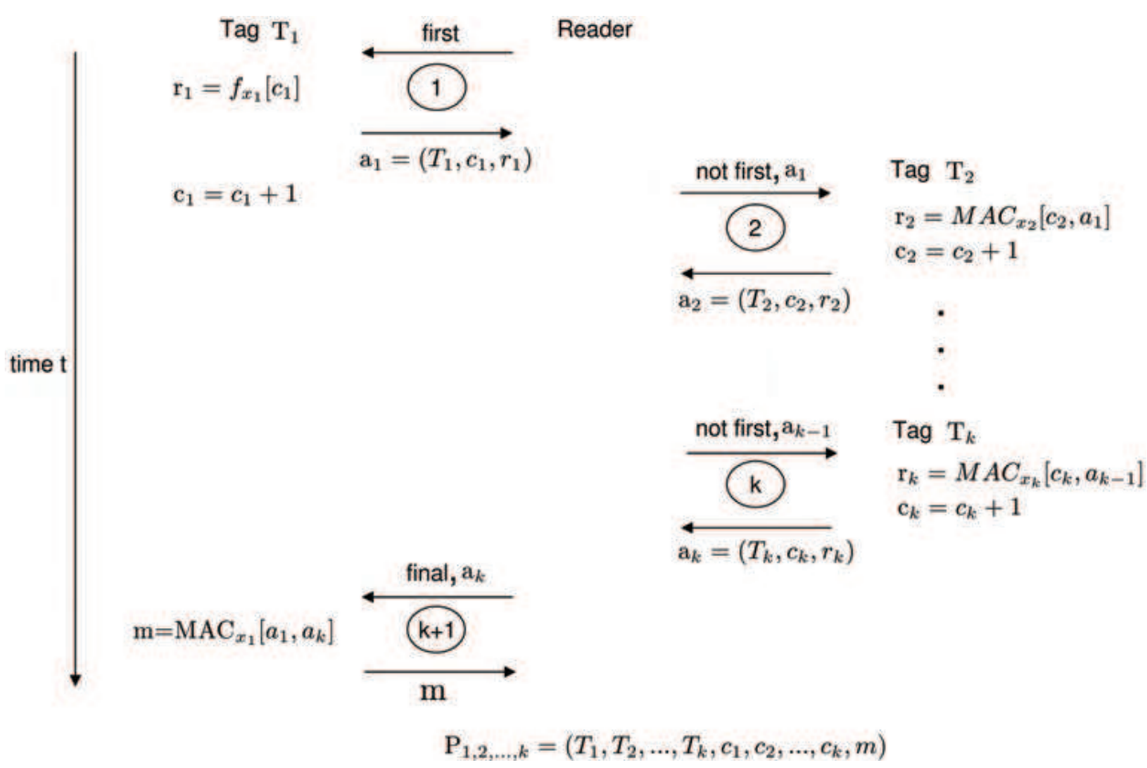

Fig. 3. Our "yoking-proof" protocol for a group of RFID tags.

Notice that each tag computes a MAC of a message that is a function of a MAC computed by the preceding tag in a yoking chain. This ensures that the reader has at most t time units to create the proof. To avoid replay attacks and allow temporal ordering of the proofs, each tag increments its counter immediately after it sends a_i to the reader. Also, observe that tags need not know how many tags participate in the "yoking-proof" protocol; instead, they only need to know when to timeout.

Note that if the first tag does not update its counter right after it sends its first message, a possibly malicious reader can create a proof P that will successfully pass through the verifier, without reading all the tags within the specified time bound t. In such a scenario, a proof can be forged as follows. The malicious reader can ask the first tag to compute a_1, then wait for t time units to elapse in order to cause T_1 to timeout, then send a_1 to T_2 to obtain a_2. Then, the reader will access T_1 for a dummy computation of a_1, and send a_2 to T_1 to obtain m, and construct a valid proof $P = (1, 2, c_1, c_2, m)$.

The basic yoking protocol of Juels, as shown in Figure 1, suffers from this problem unless the counter on the first tag is incremented on a timeout, but this is not specified in Juels's paper. The security of our scheme hinges on the probability δ that an adversary A is able to construct a "yoking proof" P_{AB}, which could fool the verifier V into reporting "success", without actually reading the IDs of all the tags involved in the protocol within t time units, as intended.

Next, we state the security property of our protocol. The proof of the theorem can be found in the author's Ph.D. thesis [Bolotnyy, 2007].

Theorem: Given random-oracle assumptions for f and MAC [Bellare and Rogaway, 1993], the success probability δ of an adversary A for a grouping protocol is bounded from above by 2^{-d} where d is the message length. □

Discussion: Our group-yoking protocol can be adapted to the "Minimalist MAC" protocol proposed by Juels. However, as discussed above, a one-time proof is not very useful given the problem assumption that the reader may be malicious.

To prevent an adversary from replaying old proofs, the verifier can store counter values of tags obtained from the latest verified-correct yoking proofs in which tags participated. A replay attack will use the counter value that is less than or equal to the last recorded counter for the tag. Storing tag counter values also allows for a temporal ordering of the yoking proofs, which may be desired. If there is more than one verifier in the system, the verifiers need to be able to share the tags' counter values to prevent replay attacks.

Having counters on-board tags allows for temporal ordering of the yoking proofs and guarantees that a keyed hash function will never be computed on the same point twice. However, such tag counters require tags to maintain a persistent state between several runs of the protocol. An alternative is to replace counter values with random numbers generated on-board the tags (the tags will need to be equipped with pseudo-random number generators and maintain secret seeds or have truly random number generators). However, if random numbers are used instead of counters, and a replay of past proofs is considered to be an attack, the verifier should store all previous yoking proofs and compare them to the new proof.

We made an assumption (see Section 2) that the reader cannot physically steal secrets from the tags. We can relax this assumption somewhat. Assuming that a malicious reader did not read one or more of the tags, and is pressed for a proof by the verifier, then the owner of the reader can try to find the "first tag" and physically steal the secret key from it, allowing the reader to complete the protocol. To prevent such an attack, each tag can update its secret key

in a forward secure manner [Bellare & Yee, 2003]. For example, a one-way hash function can be used to update a tag's key at the time of a tag's counter update. The old key is then securely discarded. For this scheme to be practical, tags should maintain counters, instead of generating random numbers as timestamps, to allow the verifier to quickly determine the secret key that each tag used for computation.

In yoking proofs it is critical to rapidly establish communication with *all* the tags, otherwise the yoking-proof can not be created within the required time interval. This issue raises the object detectability problem [Bolotnyy & Robins, 2005] [Bolotnyy & Robins, 2007a]. Note, multiple *connected* tags, or *multi-tags*, can help address the object detection problem [Bolotnyy & Robins, 2005] [Bolotnyy, 2007].

6. Anonymous yoking

Observe that Juels's "yoking proof" protocol and our generalization do not hide the identities of tags – each tag sends its identifier and its counter value to the reader in the clear. Before the execution of the protocol, the reader is unaware of the identities of tags, and running the yoking protocol will reveal them to it. In some practical scenarios, we would like to preserve the identity of objects associated with the tags (i.e. not reveal tag IDs to untrusted readers). We therefore introduce a new problem formulation, called *anonymous yoking*, which in addition to the requirements of a "yoking-proof" problem, requires tags to preserve their privacy.

The protocol that we develop for anonymous yoking is very similar to the generalized "yoking proof" protocol discussed above, yet certain differences and details are important. Let $f : \{0, 1\}^d \times \{0, 1\}^* \rightarrow \{0, 1\}^d$ be a keyed hash function. Upon the reader's request, each tag will generate a random number r, and compute $a = f_x(r, value)$, where x is a secret key stored on a tag and *value* is an output of the previous tag in the chain, as in our generalized "yoking proof" scenario above. The first tag sets *value* equal to 0. Each tag will respond to the reader's request by sending (r, a) to it, and the first tag will close the chain. The detailed algorithms for the reader, a tag, and the verifier can be found in the author's Ph.D. thesis [Bolotnyy, 2007]. The proof of security is very similar to the one for the generalized protocol [Bolotnyy, 2007]. This anonymous yoking-proof protocol is privacy preserving in the Strong Privacy model of [Juels & Weis, 2006].

Note that the process of determining the tags' identifiers and verifying the proof are combined in the verification steps. The verifier will try to determine which secrets were used to compute each a_i given r_i. Since a_i is a function of a_{i-1} for all $2 \leq i \leq k$, the process of determining the secrets and verifying the proof coincide. To determine the tags' identifiers and to verify the protocol, it will take the verifier $O(k \cdot n)$ time where n is the total number of possible tags and k is the number of tags participating in the protocol. The running time can be further reduced to $O(k \cdot log(n))$ if the approach suggested by Molnar et al. [Molnar and Wagner, 2004] is used, namely arranging the tags at the leaves of a tree and associating a secret to each edge in the tree. Each tag stores all the secrets on the path from the root to the leaf where it is located. Instead of sending (r_i, a_i) to the reader, each tag will send r_i, $a_i^1 = f_{s_1}(r_i, a_{i-1}), \ldots, a_i^{\text{tree-depth}} = f_{s_{tree-depth}}(r_i, a_{i-1})$ where $s_1, \ldots, s_{\text{tree-depth}}$ are secrets from the root of the tree to the leaf where a tag is located. This algorithm modification will allow the verifier to determine a tag's identity in $O(log(n))$ time by finding the right path from the root to the leaf. However, this speedup in tag identification suggested in [Molnar & Wagner, 2004] comes at

a cost, as leakage of secrets of one or more tags poses a privacy threat to other tags. The extent of the threat is analyzed in [Avoine et al., 2005].

Observe that anonymous yoking allows an adversary or even an innocent transient entity to inject an arbitrary ("yoking-proof" supporting) tag into the "yoking-proof" construction. If the verifier does not possess the injected tag's secret key, this action will prevent the verifier from verifying the complete proof, thus causing a denial of service attack.

7. Speeding up the yoking protocols

The run-time of the "yoking-proof" protocol is the sum of the time to perform encryption, the time to establish a reliable reader-to-tag communication channel, and the time to decode the reader requests and to encode the responses. We estimate that the establishment of the communication channel, and the decode and encode times take about 1ms per tag, assuming about 1,000 security-free tags can be identified per second. The majority of the protocol time will be spent on encryption operations. Therefore, we concentrate on encryption operations for the protocol run-time analysis. The tag that starts and closes the yoking chain performs 2 encryptions and the tags in the middle of the chain perform 1 encryption. Therefore, to yoke k tags requires $k + 1$ encryptions. The time to perform a single encryption depends on the frequency of a tag/reader communication, the communication distance, the communication standard, the power consumption of a tag, the cryptographic algorithm, and the length of the secret key.

For example, for high RFID communication frequency of 13.56MHz, ISO/EIC 18000 compliant standard, and 128-bit secret key AES implementations suggested by [Feldhofer et al., 2004] with 15μA current for AES module and 100KHz AES module clock frequency, one encryption takes about 10ms. So, under these conditions we can yoke ~ 36 tags and remain within the FCC required 400ms communication window ($36 \cdot 10\text{ms} + 36 \cdot 1\text{ms} < 400\text{ms}$). There may be applications where over 40 tags need to be yoked, or where a novel minimalist RFID encryption takes long computation times, or scenarios with real-time performance requirements (i.e., multiple yokings per second). For these types of applications, the "yoking-proof" protocol should be sped up.

The yoking-proof creation can be sped up by splitting the circular chain of dependent MACs into a group of arcs, where each arc consists of a sequence of dependent MACs, and where the adjacent arcs are inter-dependent. Each arc has a single element that plays the role of the "first" and the "last" tag. Let $ID_1, \ldots, ID_k$ be the tags' identifiers sorted by the tags ID, or by the random numbers generated on-board the tags for anonymous yoking. We split the sorted list of identifiers into the desired number of groups g (e.g., $ID_1, \ldots, ID_{i_1}, ID_{i_1+1}, \ldots, ID_{i_2}, \ldots, ID_{i_g}, \ldots, ID_k$).

For example, ID_1 is the "first" and the "last" element of the first arc. It starts the chain of its group ($ID_1, \ldots, ID_{i_1}$) as described in the generalized "yoking-proof" protocol, and it closes the chain of the group $ID_{i_g}, \ldots, ID_k$. In other words, the first element of each arc starts the chain of the arc and closes the chain of the preceding arc (see Figure 4).

Note that the protocol requires multiple readers or a single reader with multiple antennas. In addition, the protocol should incorporate a medium access control scheme that allows the reader(s) to communicate with more than one tag at a time to avoid/minimize tag response collisions. The time to create the yoking proof is the sum of the reader communication times with the tags belonging to the longest arc. Therefore, the overall speedup factor resulting from partitioning the tag set into arcs, can ideally approach the number of arcs.

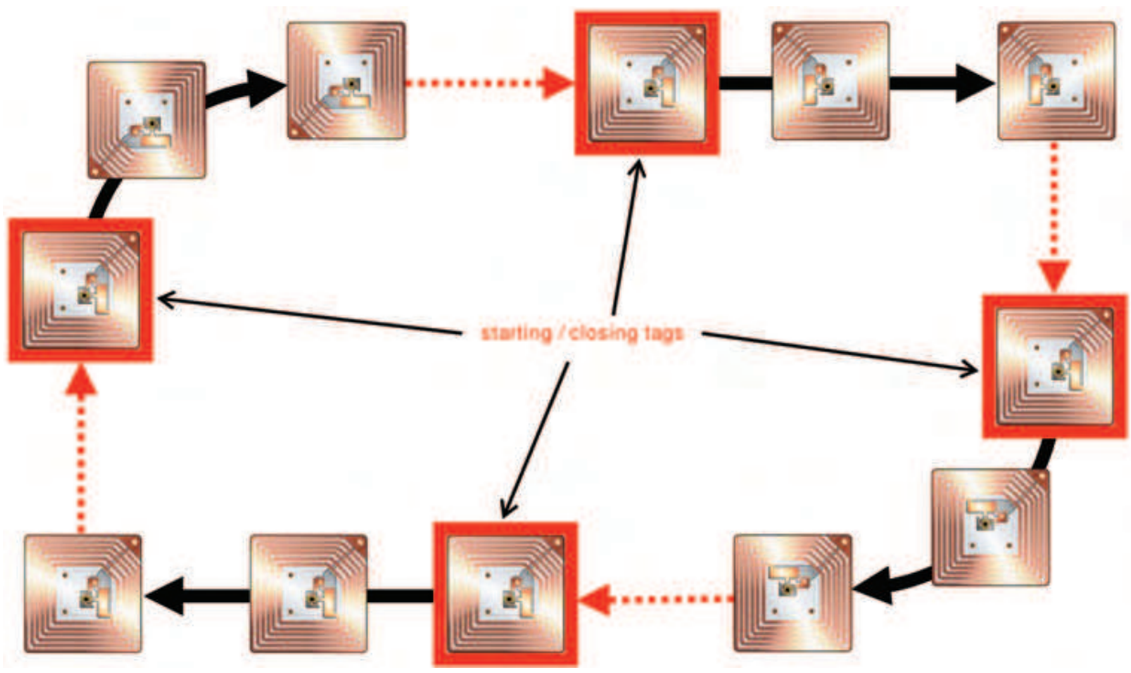

Fig. 4. Group "yoking-proof" protocol speedup. The circular chain is split into 4 arcs. Starting/closing tags are clearly marked and directions of the chain creation are shown with bold arrows. Dashed arrows represent closing operations with the source tag of the arrow proving the input to the destination tag.

The suggested speedup does not effect the security statement of the theorem for generalized yoking-proof protocol given above. The proof is also very similar and therefore omitted. The main difference in the proof is that there is more than one starting and closing operations.

7.1 Message authentication code implementations

A message authentication code (MAC) aboard a tag can be implemented using a standard cryptographic hash-based MAC (i.e., HMAC). Alternatively, Lamport's one-time signature scheme [Lamport, 1979] can be used, as noted in [Juels, 2004], where each bit position of the signature has two associated secrets - one for 0 and one for 1. The signature is an ordered sequence of secrets that correspond to 0 or 1 in each bit position. However, this scheme requires a prohibitably large memory on-board a tag. A more "minimalistic" approach, where each secret is just a single bit, was suggested by Juels [Juels, 2004]. To avoid simple forgeries, he suggests lengthening the message size, and making the message space sparse [Juels, 2004]. In our research [Bolotnyy & Robins, 2007b] [Bolotnyy, 2007] we showed how MACs can be implemented using physical hash functions (PUFs) that require an order-of-magnitude less hardware to implement than standard cryptographic hash functions. In addition, PUF-based MAC implementations allow messages to be signed more than once. Still, due to some restrictions of PUF-based MAC (limited message space or required tag presence during the signature verification), applicability of PUF-based MAC to yoking proofs may be limited.

The signatures of PUF-based MAC constructions can be long. To reduce the length of the signatures and to make them conform to the input length of each tag, a reader can apply a publicly known collision resistant one-way hash function to each MAC. The verifier will need to apply the same hash function to the signatures when verifying proofs.

8. Inter-tag communication

"Yoking-proofs" provide a good example of passive tags communicating with each other through the reader (Figure 5). In [Juels, 2004], inter-tag communication is performed as part of a larger protocol, and it did not receive enough attention as a general technique in its own right. We have found no literature that specifically discusses tag-to-tag communication between powerless tags. In the few sources where the term "tag-to-tag communication"[2] is mentioned, it refers to active (battery-powered) tags that can communicate with each other directly.

Fig. 5. Inter-Tag communication. Passive tags can communicate with each other through the reader.

In almost all RFID systems discussed in the literature, readers query RFID tags for their IDs or for some other data, and pass that data to the back-end server for processing. We envision a different paradigm of RFID systems where RFID tags, even passive or semi-passive ones, can communicate amongst themselves, using the readers as intermediaries. Such inter-tag communication capability can create additional heterogeneity in ubiquitous computing, where powerful wireless devices (e.g. readers) are able to initiate communication on their own, and communication-powerless devices (e.g. semi-passive tags, passive tags) can still communicate with their peers through powerful devices. Inter-tag communication creates opportunities for new RFID applications, but it also creates new security challenges (e.g., securing the communication channel between the tags even if "connecting" readers are untrusted, and maintaining the integrity of the transmitted data.)
Next, we give examples of applications where inter-tag communication can be beneficially utilized.

8.1 Example 1: battery-free sensing

In [Philipose et al., 2005] the authors provide evidence that battery-less tags can be used for sensing by harvesting their energy from the readers using RFID technology. Consequently,

[2] For example, [Brooke, 2005] contains the term "tag-to-tag communication" without explicitly referring to passive tags, the assumption being that the tags in question are active.

inter-tag communication can allow passive tags to share sensing data, and use it to enable future sensing activities, just like in wireless sensor networks.

8.2 Example 2: tags as mailboxes

We envision a scenario where off-line readers can exchange messages by using tags as "mailboxes". Here, one reader can send a message to another reader by writing a message onto an appropriate tag, and this message will later be retrieved by the recipient reader. Tags can also be used as proxies, permitting readers to send and receive data from tags that are outside their reading range. The data will propagate from one tag to another tag which is closer to the destination. Essentially, RFID tags can serve as powerless distributed storage devices. In this application of inter-tag communication, readers must verify the data that they read off the tags to ensure that it is not a virus (e.g., see attacks discussed by Rieback et al. [Rieback et al., 2006]).

8.3 Example 3: centralized authentication

The tag population can be partitioned into groups, with specific tags designated as *group leaders*. A group's leader is in charge of reader authentication and access control to data stored on-board all tags belonging to a group. An example of centralized authentication is a pallet tagged with a group leader tag, and individually-tagged items within the pallet serving as the group members. Individual tag access is authorized by a group leader (a powerful tag), which contains access policy information that can be updated over time. One of the benefits of such an approach is the centralization of group policy information, which allows for faster and more uniform policy updates. In addition, the tag complexity can be reduced by placing major functions on a single (group leader) tag.

Next, we describe the protocol for this centralized authentication scenario. Let k be the number of tags in the group, excluding the group leader. Let $f, h : \{0, 1\}^d \times \{0, 1\}^d \rightarrow \{0, 1\}^d$ be two pseudo-random functions. Let s be a secret key shared by the group leader and the reader, and for $i = 1, ..., k$ let s_i be the secret keys of tags T_i, known to the group leader. At the beginning of the protocol, the reader sends an access request to tag T_i and the tag responds with a *nonce* (i.e., a random "number used once"). The reader will then ask the group leader for an access certificate to tag T_i, providing it with the nonce received from the tag. The group leader will authenticate the reader using a challenge-response protocol and issue a certificate c based on the tag supplied nonce, if allowed by the access control policy. The tag will verify that the access certificate is valid and grant data access to the reader. The general form of the reader and group leader communication protocol is shown pictorially in Figure 6, and the algorithms for the reader, a tag, and the group leader are given in the author's Ph.D. thesis [Bolotnyy, 2007].

Tag data access control policy can be group-based or tag-based (i.e. the identity of a tag is not a secret, but access to the data stored on-board the tag is granted only to authorized readers). If the policy is tag-based, the reader will need to specify to the group leader the identity of the tag whose data it wants to access. Note that this scenario reduces the number of secrets that a reader needs to possess in order to collect information from the tags, since the reader only needs to share secrets with group leaders.

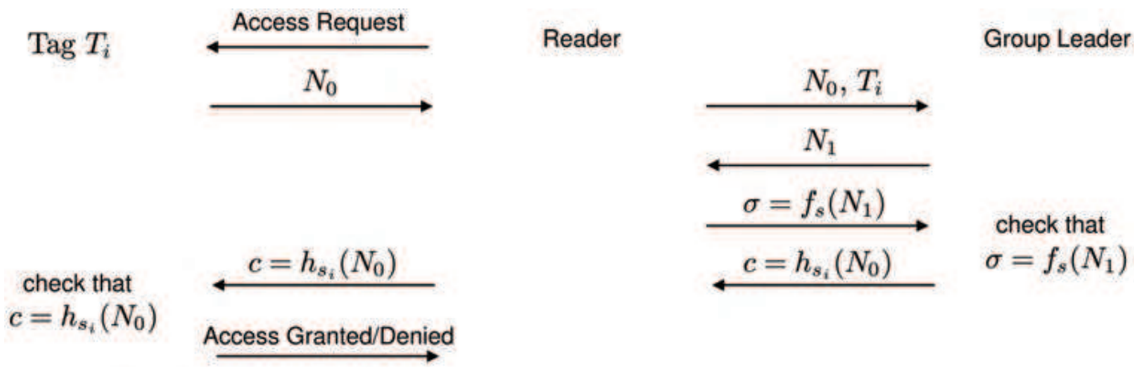

Fig. 6. Centralized reader authentication. The group leader issues a certificate to the reader, allowing access to tag T_i. Here f is the pseudo-random function that the reader and the group leader use for authentication, and s is the secret they share; h is the pseudo-random function that the group leader and the tags within a group use for authentication; and s_i is the secret that the group leader and tag T_i share.

8.4 Example 4: distributed access control

Inter-tag communication can also be used for distributed access control. A master tag grants the reader access to a resource only if all of its subordinate tags first authorize access to all dependent resources (i.e. all dependent *sub-resources* have to be acquired first, before the main resource itself is acquired). Examples of distributed access control include "safe deposit box opening" that requires multiple keys, and a system of doors where access to a door depends on successful access to a group of other doors.

Distributed Access Control Problem: A reader seeks to gain access to information, data or facility controlled by a master tag. To gain access to the master tag, the reader first needs to obtain recent authentication certificates from all subordinate tags, (i.e., all proofs must be generated within some time window t).

To solve this problem, the reader can perform an authentication algorithm with each subordinate tag based on a nonce provided by the master tag, and then present the collected authentication certificates to the master tag for verification. The master tag can time the reader, as in yoking-proofs, to verify the timely presentation of proofs.

However, if we also require the proofs to be verifiable at any time after they have been generated, the problem becomes more difficult. To provide such a proof, which joins together proofs from a number of tags, we can use a modified version of our anonymous "yoking-proof" solution described above. The reader will generate a proof, showing that it read all tags within time t, and then present this proof to the master tag. The "yoking-proof" will need to be augmented to allow for preliminary authentication, and timed from the start of the authentication rather than from the yoking construction. The master tag will then verify the validity of the "yoking-proof".

Another application of inter-tag communication is subordinate reader authentication / authorization. A group leader tag provides access to the reader or performs reader

authentication if and only if the reader has successfully authenticated itself to other tags. To gain access to the main object/data, the reader needs to prove to the guardian tag that it has successfully authenticated itself to the subordinate tags. To do so, the reader authenticates itself to each subordinate tag and provides a proof to the group leader that it was successfully authenticated by all tags. Timing is important here as we do not want a reader to present an "old authentication" or an authentication it stole by eavesdropping on communication by another reader.

8.5 Example 5: location access control

Yet another application of inter-tag communication is location access control. For example, several readers want to gain access to a resource if all of them are present at specific locations. Each reader can prove its proximity to a corresponding tag using a distance-bounding protocol [Sastry et al., 2003] [Hancke & Kuhn, 2005], and the authentication result is then shared among the tags. Access to the resource is granted if the protocol completes successfully.

The above list of examples does not exhaust all possible applications of inter-tag communication, and future research can focus on finding new applications.

9. Conclusion

We reviewed the basic "yoking-proof" protocol for a pair of tags suggested by Juels and discovered a critical omission in his protocol. We also described weaknesses in attempts by other researchers to generalize the protocol to a yoking-proof problem. We designed a protocol that creates a proof that an arbitrarily large group of RFID tags are read within a given time bound. This protocol generalizes the basic "yoking" protocol by Juels. The yoking-proof is improbable to forge and it is verifiable off-line by a trusted verifier. We modified the security requirements of the yoking-proof problem, requiring the system to maintain privacy, and gave an algorithm for this new anonymous yoking problem formulation. We also described a way to speedup our "yoking-proof" protocols.

We briefly discussed viable low-cost message authentication code (MAC) implementations. MAC implementations relying on physical unclonable functions (PUFs) require less hardware resources than the known cryptographic implementations of keyed hash functions and therefore, may be applicable to some applications of yoking-proofs. We proposed a new paradigm of inter-tag communication between passive RFID tags where tags communicate with each other through the reader. We put forward a number of interesting applications of inter-tag communication such as battery-free sensing, and distributed access control, among others. We suggested another type of "yoking-proof" for distributed access control in RFID, and described possible solutions.

The cost of tags capable to implement our generalized yoking proof protocol is likely to be greater than the cost of a basic RFID tag that only replies to the reader with a constant tag identifier. The tag cost may be even higher if in addition to the hash function, a PUF circuit is added to the tag to provide physical protection of the tag key(s) and counter value. Also, due to protocol timing constraints and the time to compute a hash value onboard a tag, it may not be possible to join all the tags within the required time period.

In this case, it is interesting to consider the use of a logical clock instead of a physical clock. Moreover, the tag capacitor charge and discharge rate may be imprecise, resulting in variable tag timeout. Since the capacitor charging rate is based on the power supplied to the tag by the reader, an adversarial reader can reduce the power supply to the first tag to a minimum in order to extend the object yoking period. Some provisions onboard a tag can be added to prevent the tag from communicating with the reader until the tag capacitor charges fully.

Future research opportunities lie in the development of new applications of inter-tag communication, utilizing the ability of tags to communicate with each other through the reader. In addition, much future work is needed in the area of PUF-based security, which may allow for low-cost implementations of yoking-proofs protocols. Inter-tag communication and yoking-proofs in particular give good examples of RFID that go beyond simple tracking of objects. Looking broadly at possible RIFD applications opens previously unexplored possibilities, thus paving the way to ubiquitous RFID deployments.

10. Acknowledgement

This research was supported by grant CNS-0716635 from the U.S. National Science Foundation.

11. References

[Avoine et al., 2005] Avoine, G., Dysli, E., and Oechslin, P. (2005). Reducing time complexity in rfid systems. In Preneel, B. and Tavares, S., editors, *Selected Areas in Cryptography (SAC), Lecture Notes in Computer Science*, volume 3897, pages 291–306, New York. Springer-Verlag.

[Bellare and Rogaway, 1993] Bellare, M. and Rogaway, P. (1993). Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. ACM Conference on Computer and Communications Security*, pages 62–73.

[Bellare and Yee, 2003] Bellare, M. and Yee, B. (2003). Forward-security in private-key cryptography. In Joye, M., editor, *Topics in Cryptology - CT-RSA2003, RSA Conference*, volume 2612, pages 1–18. Lecture Notes in Computer Science.

[Bolotnyy, 2007] Bolotnyy, L. (2007). New Directions in Reliability, Security and Privacy in Radio Frequency Identification Systems. Ph.D. thesis, University of Virginia.

[Bolotnyy and Robins, 2005] Bolotnyy, L. and Robins, G. (2005). Multi-tag radio frequency identification systems. In *Proc. IEEE Workshop on Automatic Identification Advanced Technologies (Auto-ID)*, pages 83–88.

[Bolotnyy and Robins, 2006] Bolotnyy, L. and Robins, G. (2006). Generalized 'yoking proofs' for a group of radio frequency identification tags. In *International Conference on Mobile and Ubiquitous Systems (Mobiquitous)*, San Jose, CA.

[Bolotnyy and Robins, 2007a] Bolotnyy, L. and Robins, G. (2007a). Multi-tag rfid systems. International Journal of Internet and Protocol Technology, Special issue on RFID: Technologies, Applications, and Trends, 2(3/4).

[Bolotnyy and Robins, 2007b] Bolotnyy, L. and Robins, G. (2007b). Physically unclonable function -based security and privacy in rfid systems. In *Proc. IEEE International Conference on Pervasive Computing and Communications (PerCom 2007)*, pages 211–218, New York.

[Brooke, 2005] Brooke, M. (2005). Common mistakes, uncommon best practices. http://www.rfidjournal.com/article/articleview/1483/1/15/.

[EPCglobal, 2006] EPCglobal (2006). Epc radio-frequency identity protocols class-1 generation-2 uhf rfid version 1.0.9.

[Feldhofer et al., 2004] Feldhofer, M., Dominikus, S., and Wolkerstorfer, J. (2004). Strong authentication for rfid systems using the aes algorithm. In Preneel, B. and Tavares, S., editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Lecture Notes in Computer Science*, volume 3156, pages 357–370. Springer-Verlag.

[Hancke and Kuhn, 2005] Hancke, G. and Kuhn, M. (2005). An rfid distance bounding protocol. In First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM), pages 67 – 73.

[Juels, 2004] Juels, A. (2004). 'yoking-proofs' for rfid tags. In Sandhu, R. and Thomas, R., editors, *International Workshop on Pervasive Computing and Communication Security*, pages 138–143, Orlando, FL, USA.

[Juels and Weis, 2006] Juels, A. and Weis, S. (2006). Defining strong privacy for rfid. Technical Report Report 2006/137, http://eprint.iacr.org/2006/137, Cryptology ePrint Archieve.

[Lamport, 1979] Lamport, L. (1979). Constructing digital signatures from a one way function. Technical Report Technical Report CSL-98, SRI International.

[Molnar and Wagner, 2004] Molnar, D. and Wagner, D. (2004). Privacy and security in library rfid issues, practices, and architecture. In *Proc. ACM Conference on Computer and Communications Security*, pages 210–219, Washington, DC, USA.

[Peris-Lopez et al., 2007] Peris-Lopez, P., Hernandez-Castro, J., Estevez-Tapiador, J., and Ribagorda, A. (2007). Solving the simultaneous scanning problem anonymously: Clumping proofs for rfid tags. In *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SECPerU)*, pages 55–60.

[Philipose et al., 2005] Philipose, M., Smith, J., Jiang, B., Mamishev, A., Roy, S., and Sundara-Rajan, K. (2005). Battery- free wireless identification and sensing. In *Pervasive Computing*.

[Piramuthu, 2006] Piramuthu, S. (2006). On existence proofs for multiple rfid tags. In In IEEE Intl. Conference on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU), Lyon, France.

[Rieback et al., 2006] Rieback, M., Crispo, B., and Tanenbaum, A. (2006). Is your cat infected with a computer virus? In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*.

[Saito and Sakurai, 2005] Saito, J. and Sakurai, K. (2005). Grouping proof for rfid tags. In *International Conference on Advanced Information Networking and Applications (AINA)*, volume 2, pages 621–624, Taiwan.

[Sastry et al., 2003] Sastry, N., Shankar, U., and Wagner, D. (2003). Secure verification of location claims. In *Proceedings of the ACM Workshop on Wireless Security*, pages 1 – 10, San Diego, CA.

Development and Implementation of RFID Technology

Edited by Cristina Turcu

ISBN 978-3-902613-54-7

Hard cover, 450 pages

Publisher I-Tech Education and Publishing

Published online 01, January, 2009

Published in print edition January, 2009

The book generously covers a wide range of aspects and issues related to RFID systems, namely the design of RFID antennas, RFID readers and the variety of tags (e.g. UHF tags for sensing applications, surface acoustic wave RFID tags, smart RFID tags), complex RFID systems, security and privacy issues in RFID applications, as well as the selection of encryption algorithms. The book offers new insights, solutions and ideas for the design of efficient RFID architectures and applications. While not pretending to be comprehensive, its wide coverage may be appropriate not only for RFID novices but also for experienced technical professionals and RFID aficionados.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Leonid Bolotnyy and Gabriel Robins (2009). Generalized "Yoking-Proofs" and Inter-Tag Communication, Development and Implementation of RFID Technology, Cristina Turcu (Ed.), ISBN: 978-3-902613-54-7, InTech, Available from: http://www.intechopen.com/books/development_and_implementation_of_rfid_technology/generalized_yoking-proofs_and_inter-tag_communication

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821