Practical Minimalist Cryptography for RFID Privacy

Marc Langheinrich, Remo Marti

Abstract— **The fear of unauthorized, hidden readouts has dominated the RFID privacy debate. Virtually all proposed privacy mechanisms so far require consumers to actively and explicitly protect read access to their tagged items – either by jamming rogue readers or by encrypting or pseudonymizing their tags. While this approach might work well for activists and highly concerned individuals, it is unlikely (and rather undesirable) that the average consumer should be outfitted with RFID jamming devices before stepping outside, or that anyone would bother pseudonymizing every can of soda they buy with a personal PIN code. Juels' "minimalist cryptography" offers a simple, yet effective identification and tracking protection based on simple ID rotation, but it requires that the corresponding mappings (i.e., from pseudonyms to real IDs) are electronically exchanged whenever a product changes hands (e.g., for buying a pack of chewing gums at a kiosk) – a rather impractical requirement. Our work extends Juels' concept in order to alleviate the need for passing ID mapping tables. Using carefully assembled sets of IDs based on the cryptographic principle of secret shares, we can create RFID tags that yield virtually no information to casual "hit-and-run" attackers, but only reveal their true ID after continuous and undisturbed reading from up-close – something that can hardly go unnoticed by an item's owner. This paper introduces the underlying mechanism of our extension to Juels' proposal, called "Shamir Tag", analyzes its tracking resistance and identification performance, and discusses deployment aspects.**

Index Terms— **RFID, privacy, Shamir, minimalist cryptography, secret sharing**

I. Introduction

Reliably controlling access to RFID tags has long been seen as the panacea to RFID privacy concerns. While the comparatively effortless readout capabilities of RFID (i.e., no line of sight needed) have been the technology's biggest trump over its predecessor, the optical bar code, they have also dominated the public debate about its greatest dangers: the alleged ability to comprehensively track and profile consumers [1], presumably against their will and without their knowledge, as well as its potential to facilitate criminal searches of personal belongings (or even entire households) with a simple sweep of a reader [2].

Clearly, disabling tags at checkout – either by executing a *kill-command* [3] that renders the tag silent to all reader requests, or by physically clipping the tag antenna [4] – provides the strongest protection against unwanted readouts. However, permanently silencing RFID tags also prevents a range of after-sales uses of tagged products, e.g., receiptless returns, automated recycling, or smart laundry machines. Supporting an explicit form of access control on tags would protect consumer privacy while still allowing secondary uses. Out of the many options for access control,[1] so far only password-based methods have seemed feasible for RFID tags.[2] Yet even though their general principle is easy enough for implementation on a tiny RFID tag, the practical use of such schemes is often challenging. This is mainly due to two factors: the employed communication protocol, and the envisioned deployment scale, which quickly render technically feasible solutions cumbersome, if not impractical for users.

In RFID communication, a reader typically does not know which tags are in its vicinity. Thus, in order to identify one or more RFID tags, it simply sends out a broadcast message asking for all tags (or all tags of a specific type) to reply. If password-protected tags would simply remain silent until the right password was used, a chicken-and-egg problem arises: in order to identify a tag, the reader

Manuscript received April 30, 2007, revised August 30, 2007. This work was partially supported by a grant from Hitachi Ltd., Japan

Marc Langheinrich is with the Institute of Pervasive Computing, ETH Zurich, 8092 Zurich, Switzerland (corresponding author, phone: +41-44-632-0688, e-mail: langhein at inf.ethz.ch). Remo Marti is with Ergon Informatik AG, 8008 Zurich, Switzerland (e-mail: remo.marti at ergon.ch).

[1] Access control denotes a scheme or system where access to information, a system, or a location can be granted based on what you know (i.e., a password), what you have (i.e., an access token), who you are (i.e., biometrics), what you do (i.e., personal traits), or where you are (i.e., your location).

[2] An excellent overview of currently proposed methods can be found in [5].

must know its password. But in order to select the correct password from its list of known passwords, a reader would first need to know which tag it is, i.e., it would need to identify it first. In some situations, this look-up could be manually performed, e.g., by having the tag's password (PIN) printed on the RFID label and requiring this PIN to be entered into a reader device prior to readout. While this would not work very well for carts of groceries or clothing, it is actually used for RFID tagged passports: in order to read out the embedded RFID tag, an optical reader first needs to read the name, date of birth, and expiration date from the *Machine Readable Zone (MRZ)*, which is then used to compute the tag's password and initiate the RFID readout [6].

Instead of completely remaining silent, tags could reply with a *pseudonymous* ID, which could then be used by the reader to look-up a tag's password (or even directly its ID) [7]. While this would hide a tag's true ID, the fixed pseudonym would still facilitate tracking, i.e., the re-identification of the same tag across several readers. Other proposals thus employ *ID-chains*, where the tag computes a new ID for each readout using a cryptographic formula [8]–[10]. As this does not allow direct lookups anymore, readers need to initiate a *key search*, where each known key is tested whether it yields a tag's reported, dynamic ID. Given the envisioned deployment scale of RFID tagged items, such schemes would quickly become unmanageable even for personal use (let alone on an industrial scale).

Juels [11] rightly points out that such "hard crypto" approaches might actually not be needed for RFID tags: in contrast to, say, internet servers, these tags are not constantly available from almost anywhere in the world through data networks, but instead require that the attacker be in close physical proximity to a tag for readout. Thus, attack models for RFID tags need to be adapted to such realities. Juels argues that a simple list of pseudonymous IDs in practice works just as well, and at much lower costs, than cryptographically computed ID-chains, which require both costly key searches and expensive RFID hardware. In his scheme, which he dubs "minimalist cryptography", Juels supposes that an adversary is only able to periodically (i.e., not constantly) scan a certain tag. Tags contain a small collection of pseudonyms and release a different one upon each reader inquiry. Authorized readers can store the full set of tags and thus quickly re-identify it, while an unauthorized reader would neither know how to map the pseudonym to a true ID, nor be able to consistently track an item due to the ever changing pseudonym. In order to prevent rogue readers rapidly reading out all available pseudonyms of a tag in a single sweep, Juels proposes to "throttle" tag replies, effectively limiting the number of pseudonyms a reader can read per second (to, e.g., one).

While Juels' scheme alleviates the need for cryptographic functions on the tag, thus lowering tag costs (and making his proposed scheme more affordable), it nevertheless requires authorized readers to obtain a tag-specific mapping table that supports translating a pseudonym into a tag's true ID. In order to allow a consumer to read out the ID of an item she just bought, this mapping table would need to be electronically transferred to her at checkout. While such data exchange might still be feasible in the industrial supply chain, i.e., between retailers and manufacturers, it significantly raises the bar for ordinary citizens and small vendors. Imagine buying a can of soda at a street cart, or a pack of coffee in your favorite neighborhood deli – not only would consumers without an electronic device of some sort be at a disadvantage, but it is questionable whether small business could afford the necessary IT-equipment needed to support such transfer [12]. As an alternative to an on-the-spot exchange, one could imagine setting up a central repository where this information could be downloaded later. However, this download would again require some sort of access token (e.g., a password or key), otherwise any attacker could trivially lookup such pseudonyms, invalidating the entire scheme.

Much more appealing are thus keyless schemes to RFID privacy, such as Juels et al.'s *blocker tag*: a specifically engineered RFID tag replies to each and all reader requests, thus causing signal collisions with all regular RFID tags in its vicinity, effectively preventing any replies to reach the reader [13]. However, in order to prevent a single blocker tag from paralyzing an entire row of checkout lanes or loading docks, the authors propose to mark the IDs of personal items that should be protected with a specific *privacy prefix*, and subsequently limit the blocker tag to only block tag readouts of this particular ID-space. This again prompts the need for some sort of access control for setting

this prefix, putting us back to square one. Fishkin et al. [14] instead propose a simple but intuitive *distance-based* access control scheme, where tags measure the noise level in a reader's signal and infer the relative distance of the reader. Based on the reader's distance, tags can then reply with different levels of detail, i.e., they can remain silent for far away readers, reveal a partial ID for closer ones, or disclose their full ID to nearby readers. While this approach would work without any explicit access-control management schemes, it hardly seems feasible in the foreseeable future, especially for low-cost passive RFID tags. Apart from the increased costs for the required on-tag circuitry to detect the signal to noise ratio, distance-based authentication might also not always yield the desired functionality, e.g., when passing narrow passageways or small store entrances.

This paper presents an alternative RFID privacy scheme that combines the hardware simplicity of Juels' "minimalist cryptography" approach with the much lower infrastructure requirements of a keyless access control model. While we also use a set of pseudonyms in each tag to prevent direct item identification and hamper traceability, our pseudonyms can be cryptographically combined to yield the true ID of an item, without the need for external mapping tables. Just as in Juels' scheme, access control is limited by the "throttled" replies of a tag, effectively limiting the information an unauthorized reader can obtain from an item. Our system thus discloses an item's ID only after a *thorough* scan, i.e., only after a reader has assembled enough pseudonyms of an item in order to reassemble the original ID. An RFID-tagged shirt, for example, could still inform a smart laundry machine of the correct water temperature after a few spins of the drum, but casually passing a reader while wearing the shirt would not allow for enough information to be read out. Our approach is based on the idea of *shared secrets* [15], in which a piece of information is broken into many smaller parts that individually yield no information about the original information.

The remainder of this paper is structured as follows. We begin in section II with describing the two concepts underlying our proposed mechanism: bit-throttling and shared secrets. Section III then describes how these can be combined into a cryptographically lightweight solution, our so-called "Shamir Tag", which effectively hampers unauthorized identification and tracking. It also offers a detailed analysis of the capabilities and limits of Shamir Tags, followed by a discussion of practical deployment issues in section IV. We close with conclusions and an outlook on future work in section V.

II. Fundamental Concepts

The main goal of our privacy-friendly RFID tag is to hide the true ID (e.g., its Electronic Product Code [16], EPC for short) of an RFID-tagged item, so that unauthorized readers cannot secretly spy out our belongings, while legitimate parties (including ourselves) can still enjoy the benefits of automatic identification. This rules out logical or physical tag deactivation solutions – our tags should eventually reply to authorized reader requests. Additionally, unauthorized readers should not be able to repeatedly read the same ID from one of our tags, even if they cannot resolve it, as they would still be able to track our movements this way.[3] Thirdly, we neither want to manage passwords nor pseudonym-tables, as any such scheme would quickly become infeasible in consumer scenarios.

As we pointed out in the introduction above, Juels' "minimalist cryptography" approach [11] fits the first two requirements nicely, without any need for costly crypto-circuitry on the tags: by simply storing a set of, say, 10 pseudonyms instead of a single true ID, only authorized readers will be able to translate a tag pseudonym into the corresponding item's EPC.[4] Additionally, by throttling tag replies, an unauthorized reader would only be able to read (on average) one of these pseudonyms each time the item is in range, thus making it much harder to repeatedly track the item. Our third requirement, however, is violated in Juels' scheme by the need for active exchange of such pseudonyms-to-ID-tables: A consumer buying a new sweater would first need to obtain the translation for the embedded RFID chip, in order to be able to use it in her smart laundry machine, or during a warranty replacement.

[3]Note that such fixed IDs would not need to be unique to allow tracking, as even classes of tags can form unique *constellations* that can act as identifiers [8].

[4]Juels actually describes a much more elaborate and cryptographically secure system in [11] – the multi-pseudonym version is simply the most basic form of privacy protection that can be implemented with his protocol. These additional elements, however, do not concern privacy protection but rather RFID security aspects, such as protection from cloning attacks.

While in theory trivial, the organizational complexity of transferring the corresponding lookup table for an item to a user-controlled system at the point of sales seems hardly feasible. One obstacle is the rate of technological adoption: each user would need to carry a compatible electronic device (e.g., an NFC-enabled [17] mobile phone) with the corresponding software installed for receiving the transfer – even assuming an optimistic rate of adoption, it would take many years until a majority of shoppers would own and regularly carry such a device. Another problem is costs: While large supermarkets could in principle outfit all their checkout-lanes with such transfer stations, these upgrades would involve significant costs without providing any benefit to retailers. Smaller shops and kiosks, however, without an integrated electronic sales system, would need to update their entire procurement, inventory, and sales operation – an investment that could easily dwarf their yearly revenues. Last but not least, unless payment is directly integrated into this data exchange, the additional overhead of performing this transfer could quickly frustrate shoppers and slow-down checkout-times, even without any technical glitches (which might require periodic updates of both sales systems and consumer software, especially during early deployment).

What is missing from this simple approach, then, is a way to encapsulate the true ID of an item into the tag itself, without requiring authorized readers to know a corresponding password or translation table. At the same time, however, it must remain reasonably hard for an attacker to read out this ID directly. Our solution is based on two fundamental concepts, *time-delayed identification* and *shared secrets*, which together fulfill our above-stated privacy goals. These will be discussed in the following paragraphs.

A. Time-Delayed Identification with Bit-Throttling

An obvious solution is to retain Juels' idea of limiting the amount of data disclosure per time frame, yet keeping the original ID of the tag intact, i.e., not replacing it with pseudonyms that would require costly translation-table management. Instead of replying with its full ID, tags would only reply with a few bits at a time, either in a well-known order or randomly (which would entail sending information about this random order back to the reader as well).

As pointed out in the introduction, however, tags need a unique ID in order to be addressable by a reader. How could the reader discern multiple tags replying with individual bits each, without mixing up two or more bits from different tags? The solution is to have each tag reply with a randomly generated temporary ID upon the initial tag interrogation. Subsequent communication between reader and tag then proceeds under this temporary ID, until the tag is powered down again, ready to create a new temporary ID upon the next activation. This protocol is known as the Q protocol, which is standard for all RFID tags implementing the so-called EPC RFID Class-1 Generation-2 UHF air interface [3] (also called "Gen-2 tags"). While Gen-2 tags use this protocol to shorten communication times,[5] we would employ it as a true temporary identifier as long as not all bits of the tag's true ID have been disclosed.

In order for the reader to be able to put the individual bits into the correct order, we will need to enclose information about their original position into the reply as well. This additional data will introduce communication overhead and thus slow down transmission rates – but this is what bit-throttling tries to achieve anyway, so we will not need to be too concerned. However, finding an efficient scheme might still be desirable, as this would help to reduce tag complexity and energy usage (during readouts), thus ultimately reducing costs.

One intriguing idea would be to re-use the temporary identifier as implicit information about the following order of bits. We simply systematically enumerate all possible bit-orderings and let our temporary identifier designate the permutation that designates the chosen order. E.g., the $3! = 6$ possible ways of sending 3 bits could be enumerated as $\{$0=(123), 1=(132), 2=(213), 3=(231), 4=(312), 5=(321)$\}$. A tag using a temporary ID of 3 would thus indicate that it would first send its 2^{nd} bit, then its 3^{rd} bit, and then its 1^{st} bit.

However, given that a 96-bit EPC would allow for $96! \approx 10^{150} \approx 2^{499}$ different orderings, one would need a 499-bit identifier to uniquely express every single order possible. This would not only require using a much larger random number generator

[5]The randomly generated Q-bit ID is only 16 bits long, making it much shorter than the tag's 96-bit EPC, thus significantly reducing communication overhead [3].

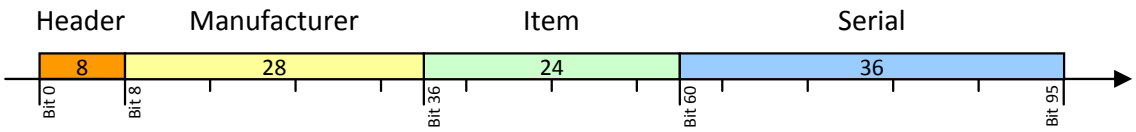

Fig. 1. *A 96-bit EPC Code (GID-96).* The EPC tag data specification [16] defines a number of data formats for an EPC label. Shown above is the "General Identifier" (GID-96), a newly defined format for identifying the manufacturer, item, and individual serial number of an object. The actual format used is described in the 8-bit header (cf. table I below).

(RNG) on the tag,[6] but also the implementation of corresponding logic circuits that could select and execute the ordered disclosure of 96 bits according to a 499-bit identifier.

Much more feasible might thus be prepending a simple 7-bit ordinal (for a 96-bit EPC) to each bit value, resulting in a sevenfold communication overhead, but greatly simplifying the process, as tags would only need to store a temporary bit mask indicating the bits already sent, and repeatedly executing the built-in 16-bit RNG for randomly determining the next unsent bit that should be transmitted. In its simplest form, bits could also be transmitted sequentially, starting from the least or most significant bit.

TABLE I

THE EPC TAG DATA FORMATS AND THEIR HEADERS [16].

Header	Code and Size
01	reserved
0010 1111	DOD-96
0011 0000	Serialized Global Trade Item Nr. SGTIN-96
0011 0110	Serialized Global Trade Item Nr. SGTIN-198
0011 0001	Serial Shipping Container Code SSCC-96
0011 0010	Serialized Global Location Nr. SGLN-96
0011 1001	Serialized Global Location Nr. SGLN-195
0011 0011	Global Returnable Asset Identifier GRAI-96
0011 0111	Global Returnable Asset Identifier GRAI-170
0011 0100	Global Inidvidual Asset Identifier GIAI-96
0011 1000	Global Inidvidual Asset Identifier GIAI-202
0011 0101	General Identifier GID-96

How effective would such a protection be? While we evaluate our proposed approach in section III more formally, we can already venture qualitative estimates here. Using a sequential (i.e., non-random) bit disclosure scheme would face two problems, as Fig. 1 illustrates: Starting from the left-most bit of an EPC-tag, this would first disclose an item's header, which defines the format of the following identifier. While most items would use the "GID-96" header 0011 0101 [16], resulting in minimal disclosure (cf. table I above), items issued by the U.S. military might use the "DOD-96" header 0010 1111, which is the only header starting with 0010, thus disclosing military-issued items after only four bits. Beginning from the other end of the tag ID first discloses an item's serial number, which limits the informational value of such data yet greatly enhances the traceability of such bits. A random bit disclosure alleviates some of these concerns, as bits can arrive in any order from any part of the item's ID, making it difficult to track items. However, since each bit must include its positional information, even very few bits could already disclose unusual bit combinations (thus increasing traceability) or even manufacturer or item data (e.g., knowing bits three and four would be enough to identify the DOD-96 header).

Hence, while bit-throttling is a priori a very simple and reasonably effective scheme to implement a password-less hard-to-track RFID tag, the potential for disclosing sensitive information with only a few bits significantly affects its protection from identification attacks.[7] We therefore need a way to keep our password-less approach, but protect the tag data better.

B. Tag IDs as Shared Secrets

Instead of keeping the true ID of the tag in plaintext, we could encrypt it so that knowing the individual bits would not directly yield information about serial numbers, manufacturers, or tag issuers (e.g., the U.S. military). This is, after all, what most RFID privacy proposals, including Juels' minimalist one, are about: they replace the true ID of an item with seemingly random information. However,

[6]Current Gen-2 tags only need a 16-bit RNG. Alternatively, one could chain the output of several runs, yet even then much larger memory banks would be required.

[7]Section III-A discusses how bit-throttling can still support item identification in industrial logistic applications.

as we pointed out above, proposed schemes for resolving such (apparent) randomness into a true ID requires passwords, online lookup services, or translation tables – thus creating an infrastructural burden.

An alternative to this would be to keep the simple approach taken by Juels, but constructing the individual pseudonyms in such a way that – once most or all of them have been read – they can be used to directly reassemble the hidden ID of the tag. Just like in our bit-throttling approach above, the information would be released slowly over time, yet instead of releasing the individual bits of the true ID (which might still disclose sensitive information), we release individual pseudonyms that can be reassembled into the true ID.

Encoding a secret number into a number of individual pieces that, taken by themselves, yield no information about the secret, is a well-known concept in the field of cryptography, and is typically called *secret sharing*. In a secret sharing scheme, each participant receives a *share* that is a part of a secret. The secret can only be recovered if enough participants cooperate in recombining their shares. A typical real-world application for such a scheme is granting access to a joint account, but only if all account holders are present (and present their shares). Schemes that allow a reconstruction of the secret with only t out of n participants involved[8] are called *(t,n)-threshold schemes*. They fulfill the following properties:

- No subset of participants smaller than a threshold t can gain information on the secret s, even when cooperating with each other.
- Any subset equal to or larger than a threshold t can reconstruct the secret s at any time.

One of the most famous (t,n)-threshold schemes was introduced by Shamir in 1979 [15]. It is based on polynomials, and in particular on the observation that a polynomial of degree $t-1$ is defined by t suitable coordinate-pairs (x_i, y_i). To encode a secret s for n participants with a threshold t, one chooses a random polynomial of degree $t-1$ that crosses the y-axis at the value of s. The n participants are each given exactly one point on the polynomial's curve, thus allowing any t members to compute the exact polynomial and thus the y-intercept s.

[8]This might be useful, e.g., for granting access to the account if already a majority of account holders is present.

The reconstruction of the secret is essentially a polynomial interpolation based on the *Lagrange* formula. Since only the y-intercept is of interest, it can be simplified to the following formula (with k being the number of shares):

$$s = q(0) = \sum_{i=1}^{k} y_i \prod_{1 \le j \le k, i \ne j} \frac{x_j}{x_j - x_i} \qquad (1)$$

In practice, computing the secret s given large numbers of shares (e.g., thousands) quickly becomes infeasible. Calculations are therefore carried out in a finite field modulo p (written as $\mathbb{Z}_p$), with p being a large prime number, equal or larger than the largest secret s that should be encoded. Not only does this reduce the size of exponents, but it also removes the need for floating point operations, as all computations are performed using integers (thus eliminating the numerical error that would occur with fractional numbers).

A comprehensive discussion of this topic is beyond the scope of this paper, but an excellent introduction, as well as efficient algorithms for solving (1) above, can be found in [18].

So how can we make use of such a secret sharing scheme in the context of RFID? Using Shamir shares, remedying Juels' simplistic (in a good sense!) pseudonym approach is straightforward. Instead of storing random pseudonyms, our RFID tags are initialized with a selected set of n shares encoding the tag's respective ID. Depending on tag memory and desired privacy level, we can use fewer or more shares to encode the tag ID: the more shares we use, the harder it will be for an attacker to track an item. Similarly, the longer it takes the tag to emit the required number of t shares (depending on our throttle rate r), the harder it will be for an attacker to identify our item.[9] Due to the basic properties of Shamir's scheme, reading out less than t tags (even $t-1$) will reveal nothing about the tag's true ID – though the active throttling should keep the chances of accidentally disclosing close to t shares rather low. However, once t or more tags have been read (i.e., after a tag has been read out continuously for a long enough period of time), any reader can compute the tag's ID, without the need for manually exchanging translation tables.

[9]Recall that the values of n and t in a Shamir scheme do not have to be secret. Thus, in order to simplify operations, an RFID Shamir scheme would simply use fixed known values for both parameters.

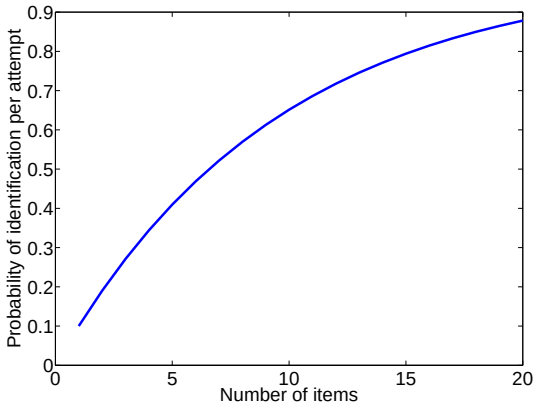

Fig. 2. *Probability of Identification.* A simple multi-ID tag with 10 shares/pseudonyms as proposed by Juels [11] is less effective when multiple tagged items are carried.

We again postpone the detailed analysis to section III below, and only assess the immediate effects of such a scheme here, i.e., the protection from tracking and identification attacks.

In contrast to our previously proposed simple bit-throttling, the use of Shamir shares effectively prevents the partial disclosure of any tag information: As long as less than t shares have been collected, no information about the encoded ID is known. Shares are given out one by one, in a throttled manner, so that no reader can read more than a few shares at once. Any such subset of shares yields no information at all about the tag's ID.

The ratio between t and n can be used to control traceability and identification: Using a large n increases the resistance to tracking, as more shares (i.e., pseudonyms) lower the chances for an unauthorized reader to re-identify our item when it passes the same reader at a later time (or a different reader that is connected to the same system). By keeping t small, we can still allow the identification of an item in a reasonable amount of time, as not all n shares need to be disclosed.

Obviously, tracking resistance degrades with the number of items we are wearing/carrying. If we have only a single tag with n shares (pseudonyms), only one in n reading attempts (on average) will results in a duplicate share being read. However, when carrying i such tags, the chances increase again that one of them replies with a previously disclosed share. The probability for a reader of finding at least one known share is simply 1 minus the probability of reading *no* matching share:

$$P(X \geq 1) = 1 - \left(\frac{n-1}{n}\right)^i \qquad (2)$$

Fig. 2 plots an example using $n = 10$ shares. For example, when carrying four such tags, (2) tells us that a reader would have a $1 - (9/10)^4 = 34.39\%$ chance of reading a previously disclosed share, i.e., on every third read attempt.

Of course, this computation assumes that each of the pseudonyms or shares is unique, which holds true for the basic scheme by Juels. Shamir shares, however, are computed separately and can very well overlap between different items. So while encountering a simple *pseudonym* twice practically guarantees that the same item has been passing the reader, the fact that the same *share* has been seen twice should not necessarily imply that it has been the same item.

In practice, however, the space of all possible shares will be very large compared to the number of actually tagged items. Recall that the large prime number p that defines the finite field in which we carry out our computation must be at least as large as our biggest secret s that we want to encode. The size of p thus defines the range of possible y-values that our shares can take. For the x-values in our computation (cf. (1)), we want to make sure that we have enough options for choosing n shares (remember that operations are carried out in $\mathbb{Z}_p$, i.e., integer space). The range of $q(x)$ should thus be at least as large as n, possibly slightly larger, especially for small values of n. The total number of possible shares would then be $p \times q$.

For example, if we want to encode a 96-bit EPC with $n = 10$ shares, our prime number p would need to be at least as big as $2^{96} \approx 8{\cdot}10^{28}$, and our range of x-values should allow for at least 10 values, better yet 100. The entire Shamir space $p \times q(x)$ would thus contain 10^{30} possible shares. This is much larger than the potential number of tagged items in the world, even when assuming hundreds of billions of items (10^{12}). The chances that any two items would by chance feature one and the same point in a space as large as 10^{30} are for all practical purposes zero (i.e., shares will be more or less unique).

Having unique shares not only makes tracking easier, it also opens up the possibility of gradually enumerating all shares of an item (this applies equally to Juels' pseudonyms) and thus eventually

identifying it (which is unique to our approach, however): As soon as an attacker is able to read not only one, but two or more shares or pseudonyms of the same item (e.g., if the throttling rate is not slow enough, or if the person lingers in front of a reader for a longer-than-expected time), these shares or pseudonyms form *ID-chains*, which can be connected (like a puzzle over time) whenever an overlapping share or pseudonym is found in a log file.

So while Shamir shares effectively protect the contents of the tag from casual observers without the overhead of any password management, they don't protect well from tracking attacks. Moreover, many short encounters can potentially be assembled into a complete set of $t \leq n$ shares, thus allowing the retroactive identification and tracking of items. Taken by itself, then, a shared secret does not seem to help much either.

III. The Shamir Tag

If bit-throttling protects from tracking attacks, but fails to protect the tag information, while Shamir shares hide tag data very well, but are open to tracking attacks – how would a combination of these two options work?

Our proposed solution to our initial problem – hiding tag information and item traces without requiring costly password management – combines the encoding of EPC data into Shamir shares with the bitwise release of information. We call such tags "Shamir Tags". Shamir Tags can be constructed for any kind of tag information s (e.g., an EPC GID-96 code), simply by creating a set of n Shamir shares that encode the secret value s, and then concatenating both, the (x, y) values of each share and the concatenated shares themselves, into a single long bit-string. Note that the order of the shares does not matter, as long as they all conform to a previously chosen format.

Fig. 3 shows the principal construction of a Shamir Tag using a practical example: An item's 96-bit EPC is considered a shared secret s, for which $t = n = 3$ shares are then computed (using, e.g., an algorithm as described in [18]). Each share is randomly drawn from a polynomial in the Shamir space $p \times q$, with p being a prime number larger than the largest secret $s = 2^{96}$ and q being large enough to allow a variety of different x values to choose from – in this example we chose $q = 1024 = 2^{10}$, though with only 3 shares, we could have used as few as 3 bits (as $2^3 > 3$). The $t = n = 3$ shares are concatenated into a single bit-string of constant length ($x_0|y_0|x_1|y_1|x_2|y_2$), which in turn is stored on the item's RFID tag instead of its true EPC. Section IV-A below discusses the choice of parameters in more detail.

During readout (i.e., after being powered by a reader's field), the tag first chooses a temporary random ID (e.g., a 16-bit RNG as described in section II-A above) in order to be addressable by the reader. The random ID does not allow tracking or identification, as it is regenerated every time the tag moves out of a reader's field. The tag then responds to an ID inquiry with a time-delayed, random (i.e., non-sequential) *bitstream*, prepending a simple o-bit ordinal in front of each payload bit in order to indicate this bit's position. This stream does not have to be uniform over time – in our example in Fig. 3 the tag immediately discloses 16 bits and only then continues to return single bits at a time, until all 318 bits have been disclosed. As soon as the reader has received enough bits to reconstruct at least t shares, it can then calculate the item's true EPC and identify it.

Note that the value of t must not be known in advance. Given a random selection of shares, a reader could in practice always verify the correctness of intermediate results by recomputing the secret s while omitting an arbitrary share from the result. If s changes, only t or fewer shares have been read. If all n shares have already been read, it is very likely that $t = n$ holds (while one might easily construct a counter example to this heuristic, the random selection of shares would make this extremely unlikely to appear in practice). However, the values of t and n might equally well be defined within an RFID-standard, as they don't constitute secret information.

The following sections analyze the feasibility and performance of this approach from three different angles. First, we need to know how well such tags will actually work for *authorized* readers, i.e., the owner of an item. Secondly, we need to analyze how *unauthorized* readers are prevented from identifying the item easily. Last but not least, the tracking protection needs to be examined. We will continue to use the construction parameters presented in Fig. 3, i.e., Shamir Tags consisting of three 106-

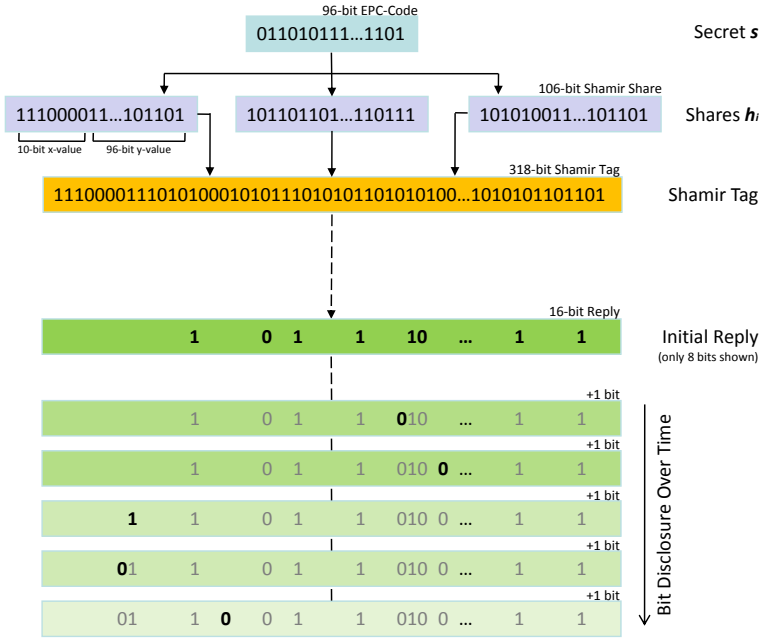

Fig. 3. *Principal Construction of a Shamir Tag.* Based on the tag's "true" ID, e.g., its EPC-code, multiple Shamir shares are concatenated to form the tag's new ID, which is then stored on the tag. Upon reader inquiry, an initial set of random bits is released, with subsequent throttled single-bit releases. Eventually, all bits will be released and only then can the original "true" ID be computed.

bit Shamir shares (total of 318 bits), though many other values are possible. Section IV-A will discuss the parameter choices in detail.

A. Authorized Identification Performance

In order for our system to be usable, it is important that applications typically do not have to wait for minutes or hours before a tag-ID can be reconstructed from the read bits. It is only the fast hit-and-run skimming attacks, as well as careless catch-all readers, that we want to prevent from identifying our tags.

The main difference that we assume between an authorized and an unauthorized party is that the former one already knows the tag, while the latter one has not seen it before. Thus, we want our system to allow private citizens or merchants to quickly identify their *own* tags, given a small list or database. This database might have been received from a supplier enumerating today's shipment, or be the result of smart laundry machines or wardrobes silently reading out whatever clothes have been put in by their owner. The idea is that there should be no *need* to transfer this information between owners in advance, but it might speed up operations, especially in the logistics domain.

The question then becomes how fast we can perform this lookup, or more precisely: How many bits do we need to uniquely identify an item from a certain known item population? Assuming that bits are uniformly distributed, about half of the population should have a `1` in a particular position, the other half a `0`. This reasoning can easily be extended to apply to multiple bit positions: each additional bit lowers the chances by 50% that items in the population will share this particular combination of bits. The number of items i in a population of B bit-strings that share a certain combination of b bits is thus

$$i = \frac{B}{2^b}. \tag{3}$$

Table II shows values of i for different bit combinations of length b and varying tag populations $B = \{100, 10^3, \ldots, 10^{10}\}$. For example, in row 20 we see that a specific combination of 20 bits from different positions in the ID would on average only be found in 0.95 items in a population of 1 million items. 20 bits should therefore be enough to uniquely identify an item within 1 million items. Since we use randomly chosen Shamir shares for the creation of our Shamir Tags, we can assume a uniform distribution of `0`s and `1`s in each tag,

TABLE II
NUMBER OF ITEMS IDENTIFIED BY BIT-COMBINATIONS OF DIFFERENT LENGTHS

Bits ↓ Items →	100	1 000	10 000	100 000	1 000 000	10 000 000	100 000 000	1 000 000 000	10 000 000 000
1	50	500	5 000	50 000	500 000	5 000 000	50 000 000	500 000 000	5 000 000 000
2	25	250	2 500	25 000	250 000	2 500 000	25 000 000	250 000 000	2 500 000 000
3	13	125	1 250	12 500	125 000	1 250 000	12 500 000	125 000 000	1 250 000 000
4	6	63	625	6 250	62 500	625 000	6 250 000	62 500 000	625 000 000
5	3	31	313	3 125	31 250	312 500	3 125 000	31 250 000	312 500 000
6	2	16	156	1 563	15 625	156 250	1 562 500	15 625 000	156 250 000
7	**0.78**	8	78	781	7 813	78 125	781 250	7 812 500	78 125 000
8	0.39	4	39	391	3 906	39 063	390 625	3 906 250	39 062 500
9	0.20	2	20	195	1 953	19 531	195 313	1 953 125	19 531 250
10	0.10	**0.98**	10	98	977	9 766	97 656	976 563	9 765 625
11	0.05	0.49	5	49	488	4 883	48 828	488 281	4 882 813
12	0.02	0.24	2	24	244	2 441	24 414	244 141	2 441 406
13	0.01	0.12	**1**	12	122	1 221	12 207	122 070	1 220 703
14	0.01	0.06	0.61	6	61	610	6 104	61 035	610 352
15	0.00	0.03	0.31	3	31	305	3 052	30 518	305 176
16	0.00	0.02	0.15	2	15	153	1 526	15 259	152 588
17	0.00	0.01	0.08	**0.76**	8	76	763	7 629	76 294
18	0.00	0.00	0.04	0.38	4	38	381	3 815	38 147
19	0.00	0.00	0.02	0.19	2	19	191	1 907	19 073
20	0.00	0.00	0.01	0.10	**0.95**	10	95	954	9 537

and thus use (3) for determining the number of bits needed to reasonably identify a Shamir Tag in a particular population of $B-1$ other Shamir Tags. Note that (3) is completely independent of the actual length of the ID bit-string. For example, in a population of 1 million items, 977 have an overlap of $b=10$ bits (see row 10) – whether the ID uses 100 or 300 bits.

Table II thus allows us to determine the minimum number of bits required to uniquely identify a personal item with high probability. Assuming 50 000 personal items, a random 16-bit "fingerprint" from a readout should typically be enough to uniquely identify an item from a list of cached entries (use the column for 100 000 items and divide by two). Using simple heuristics, even millions of items could be reliably identified with such a 16-bit fingerprint. For example, commercial IT-systems would be able to roughly know which goods should be where and at what time (e.g., goods that had already been put on the shelves would not be expected at the loading dock), which would allow merchants to identify their goods directly from the initial 16-bit substring.

B. Protection from Unauthorized Identification

If identification is thus instantaneous for owners, why should others have a more difficult time? Obviously, if the attacker has no known list of items to choose from (i.e., a lookup-table that lists all personal items, or a list of today's shipment), identifying a 318-bit item (as in our example in Fig. 3) with only 16 bits is impossible. Since the true ID of an item is not stored, but only a random set of Shamir shares, there is also no danger of leaking a particular piece of information, e.g., the manufacturer or the country of origin, from such a bit-combination.

Even if an attacker would have access to such a lookup-table, identification remains difficult, given the potential size of the surveyed set. For example, if a clothing store would remember all the items it ever sold, it would still not be able to re-identify customers with sufficient certainty, as new customers with "similar looking" items might also enter the store. Assuming as few as 10 million items in circulation, a bit-combination of 16 bits would already yield an uncertainty of 153 items (see row 16 of table II) – with or without the help of a look-up table. While 10 million sounds like a lot, note that even a relatively small city of of 20 000 citizens would only need 500 items per person – a value that is easily reached within several weeks, even if only a fraction of all groceries and clothes would be tagged. Taking a small metropolitan area with 1 million citizens and as many as 1000 items per person, we would quickly reach 1 billion tags. If we assume 10 000 tagged personal items (still well below what an *authorized* party could reliably identify, cf. section III-A above), we get over 10 billion items that one would need to choose from,

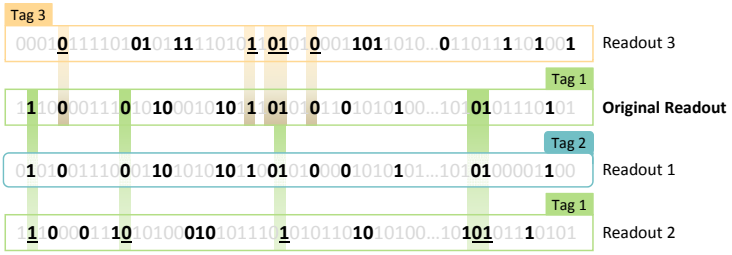

Fig. 4. *Examples of Bit-Overlaps.* Given an original readout of tag 1, subsequent readouts might share the same 16-bit readouts yet be from different tags. If fewer bits overlap (readouts 2 and 3), even more uncertainty remains.

with only 16 bits known. E.g., a smart bomb that would be programmed to detonate when a certain known tag would pass (this is a popular scenario in the debate surrounding ePassports) would get more than 150 000 false positives (see row 16, last column, in table II).

C. Protection from Unauthorized Tracking

Section II-A above already pointed out the advantages of bit-throttling for preventing tracking attacks: in order to track a single tag across multiple readers, an attacker would need to repeatedly get the same bit-combination, in order to associate these readings with one and the same tag. As we pointed out above, knowing a subset of 16 bits in a population of, say, 10 billion items still leaves a margin of error of over 150 000 items. Fig. 4 gives such an example in "readout 1": although it overlaps with the original readout in all 16 bit positions, both readouts are from different tags (tag 1 and tag 2).

Note, however, that in order to associate two different 16-bit-combinations with each other, they must actually be from the *same* 16 bit positions. So the above margin of error only applies if two readings would actually return, say, bits 0–15 in both cases. Much more likely would be that the first reading would return, e.g., bits 0–15, while the second reading would return some other bits, maybe 10–25 – yielding an overlap of only 5 bits, and thus increasing the margin of error to over 312 million items (cf. line 5, last column, table II). An example is shown in Fig. 4: readouts 2 and 3 both share a 5-bit overlap (underlined) with the original readout, yet from this information alone it is impossible to discern tag 3 from the previously read tag 1.

In order to properly assess the tracking protection that our proposed scheme offers, we thus have to compute the *average number of overlapping bits* that occur between any two readouts. In our example scheme described in Fig. 3 above, two short readouts could in theory overlap in up to 16 bits.[10] In practice, however, much lower overlaps are to be expected. The expected number of overlapping bits between any two readouts can be computed using the *hypergeometric distribution* (see, e.g., [19]), a standard means for computing the probability that a draw of N balls from a set of n black and m white balls will draw k black balls:

$$P(X=k) = \frac{\binom{n}{k}\binom{m}{N-k}}{\binom{m+n}{N}} \tag{4}$$

In our particular case, we would draw $N = 16$ balls (i.e., bit positions) from a set of $m + n = 318$ balls (cf. Fig. 3) total, which contains exactly $n = 16$ black balls (i.e., the 16 previously returned bit positions). The probability that we will find an overlap of k bit positions can thus be computed as follows:

$$P(X=k) = \frac{\binom{16}{k}\binom{318-16}{16-k}}{\binom{318}{16}} \tag{5}$$

If we want to compute the probability that we have *at least* an overlap of k bit-positions, we need to sum up (5) for all overlaps equal or greater than k. Our answer is thus

$$P(X \geq k) = \sum_{j=k}^{16} \frac{\binom{16}{j}\binom{318-16}{16-j}}{\binom{318}{16}}. \tag{6}$$

This distribution is listed in the first three columns (labeled "# matching bit positions for 1 tag") of

[10]The 16-bit number applies if we look only at the first bulk set of returned bits, cf. Fig. 3. Obviously, if tags stay longer in the vicinity of a reader, more bits might be read out on each tag.

TABLE III

PROBABILITIES OF OVERLAPPING BIT-POSITIONS IN REPEATED READOUTS (16 BIT, 10 TAGS, EXACTLY X TAGS MATCH)

	# matching bit positions for 1 tag			exactly ... tags match x or more bits from previously read set of tags											expected
x	exactly	Log (10^x)	at least	0	1	2	3	4	5	6	7	8	9	10	value
0	42.8838%	-0.3677	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	10.000000
1	38.2518%	-0.4173	57.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.21%	99.79%	9.997897
2	14.9421%	-0.8256	18.86%	0.00%	0.00%	0.00%	0.00%	0.04%	0.38%	2.22%	9.00%	23.93%	37.70%	26.72%	8.763724
3	3.3779%	-1.4714	3.92%	1.83%	9.00%	19.93%	26.14%	22.51%	13.29%	5.45%	1.53%	0.28%	0.03%	0.00%	3.297654
4	0.4921%	-2.3079	0.54%	57.93%	32.50%	8.21%	1.23%	0.12%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.531223
5	0.0487%	-3.3124	0.05%	94.91%	4.97%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.052116
6	0.0034%	-4.4732	0.00%	99.65%	0.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.003533
7	0.0002%	-5.7851	0.00%	99.98%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.000170
8	0.0000%	-7.2481	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.000006

TABLE IV

SAME AS TABLE III, BUT "AT LEAST" X TAGS MATCH

	at least ... tags match x or more bits from previously read set of tags									
x	>=1	>=2	>=3	>=4	>=5	>=6	>=7	>=8	>=9	10
0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.79%
2	100.00%	100.00%	100.00%	100.00%	99.95%	99.58%	97.35%	88.35%	64.42%	26.72%
3	98.17%	89.17%	69.24%	43.10%	20.59%	7.30%	1.85%	0.32%	0.03%	0.00%
4	42.07%	9.56%	1.36%	0.13%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%
5	5.09%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
6	0.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
7	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

table III. For example, there is a 42.8838% chance that the second readout (the "draw" of the balls) will have *no* overlap with the set of bit positions read out the first time (i.e., the 16 black balls marked in the total set of 318 balls), and a 38.2518% chance that these two readouts overlap in exactly 1 bit position (cf. rows 0 and 1 in table III). Column 2 simply shows the logarithmic value of the first column, in order to accommodate the small numbers. The third column finally gives the cumulative numbers described in (6), i.e., the probability that *at least* this many bit positions overlap (obviously, there is a 100% probability that there is an overlap of 0 bits or more). Row "x=2" can be read as: "There is an 18.86%-chance of finding an overlap of 2 *or more* bit positions". Looking at row 8, we can find that there is only a rather small chance of actually reading the same combination of 8 bit-positions twice (about 10^{-7}); for bit-combinations of length x=12, e.g., this would be as low as 10^{-14}.

Using standard combinatorics (see, e.g., [19]), we can also compute the *expected value* $E(X)$ of our hypergeometric distribution, i.e., the expected number of overlapping bit positions between two readouts:

$$E(X) = n \cdot \frac{M}{N} = 16 \cdot \frac{16}{318} = 0.805 \qquad (7)$$

This means that the returned bit-positions from a newly read tag will overlap, on average, in only 0.8 positions from a previous tag. This sounds like a very good result, as the discriminatory power of 1 or 2 bits is very low in pretty much any tag population. However, as we have discussed in section II-B above, people might carry more than one tagged item, thus increasing the chances of finding overlapping readouts (cf. Fig. 2). We thus need to adjust our above analysis to take *multiple* tags into account.

D. Multi-Item Tracking

While the expected number of overlapping bit positions between two readouts is as low as 0.8, the fact that a person will usually carry more than one tagged item will create more chances that overlapping bit positions can be found. Simply multiplying the expected number of overlapping bit-positions by the number of tags is of course not correct. Our goal is to compute the probability that j or more tags of a readout have an overlap of k or more bit-positions with a set of i previously read tags. The results are given in table IV above – subsequently we provide a step-by-step computation of these numbers.

Recall the above probability $P(X \geq k)$ as described in (6). It can be expressed as follows:

P_k : Probability that 1 item has overlap of k or more bit-positions with 1 previously read item.

$\overline{P}_k$: Probability that 1 item has overlap of less than k bit-positions with 1 previously read item.

In order to extend this computation to multiple items, we first try to find the probability that our recently read bit-combination matches one *or more* previously read items. We can express this probability Q as follows:

$Q_{i,k}$: Probability that 1 item has overlap of k or more bit-positions with 1 *or more* previously read items, given i items.

$\overline{Q}_{i,k}$: Probability that 1 item has overlap of k or more bit-positions with *no* previously read item, given i items.

We can easily compute $\overline{Q}_{i,k}$, as this is simply the probability that we find *no* overlap of k or more bits (i.e., $\overline{P}_k$) in i consecutive experiments (one experiment for each of the i items):

$$\overline{Q} = \overline{P}^i = (1-P)^i \tag{8}$$

Knowing $\overline{Q}$, we can now compute Q:

$$Q = (1-\overline{Q}) = 1-(1-P)^i \tag{9}$$

With $Q_{i,k}$ we thus know the probability that a newly read-out tag has an overlap of k or more bit-positions with i or more previously read items. However, as the current readout also comprises more than 1 tag, we will need to extend this again to allow *multiple* readouts to match i or more previously read items. We begin again by expressing this probability, let us call it R, formally:

$R_{i,k}$: Probability that 1 *or more* items (out of i items) have an overlap of k or more bit-positions with 1 or more previously read items (out of i items).

$\overline{R}_{i,k}$: Probability that *none* of i items has an overlap of k or more bit-positions with 1 or more previously read items (out of i items).

Again, computing the inverse probability $\overline{R}$ is fairly easy, as it is simply the probability of a single tag not matching 1 or more previously read items (which we called $\overline{Q}$) for i consecutive trials:

$$\overline{R} = \overline{Q}^i = (1-P)^{i^2} \tag{10}$$

This observation allows us to compute R directly as

$$R = (1-\overline{R}) = 1-(1-P)^{i^2}. \tag{11}$$

It is not trivial to generalize from this equation, i.e., to not only get the probability that *one* or more items have an overlap, but j or more. We again begin by formally describing the probabilities that we are looking for:

$S_{i,j,k}$: Probability that j or more items (out of i items) have an overlap of k or more bit-positions with 1 or more previously read items (out of i items).

$\overline{S}_{i,j,k}$: Probability that *less than* j items have an overlap of k or more bit-positions with 1 or more previously read items (out of i items).

We first observe that "less than j items" simply means "0 items, 1 item, 2 items, ... or $j-1$ items". We can thus simply sum up the probabilities for these different cases (i.e., "0 items match", "1 item matches", etc.) in order to compute $\overline{S}$. While the probability that no item matches is simply $\overline{R}$ (cf. (10)), we need yet another probability, let us call it T, that describes how *exactly* j items match k bits or more out of the previously read i items:

$T_{j,i,k}$: Probability that *exactly* j items (out of i items) have an overlap of k or more bit-positions with 1 or more previously read items (out of i items).

Computing $T_{j,i,k}$ is straightforward. The probability that exactly 1 item matches (T_1) is simply the probability that only item 1 matches, plus the probability that only item 2 matches, etc. We already computed the probability that a single item matches k bits from a pool of i items – recall the definition of $Q_{i,k}$ above. The probability that out of i items only a single one matches, is simply i experiments in a row with only one success (i.e., Q) and $i-1$ failures (i.e., $\overline{Q}$). As we have i such items, and each one could be the single matching item, we have to multiply this probability by i:

$$T_1 = i \cdot Q \cdot \overline{Q}^{i-1} \tag{12}$$

For the case "exactly 2 items match" (T_2), we have 2 successes and $i-2$ failures. However, there are many more combinations for which particular two items match – namely $\binom{i}{2}$. T_2 is thus

$$T_2 = \binom{i}{2} Q^2 \cdot \overline{Q}^{i-2}. \tag{13}$$

We can generalize this formula to T_j as follows:

$$T_j = \binom{i}{j} Q^j \cdot \overline{Q}^{i-j} \tag{14}$$

We can now compute $\overline{S}_j$ and with this S_j as follows. Recall that $\overline{S}_j$ was simply the probability that no item matches ($\overline{R}$) plus the probability that one item matches (T_1), etc., up to a match of $j-1$ items:

$$\overline{S}_j = \overline{R} + \sum_{m=1}^{j-1} T_m \tag{15}$$

$$\begin{aligned} S_j = 1 - \overline{S}_j &= 1 - \left(\overline{R} + \sum_{m=1}^{j-1} T_m \right) \\ &= 1 - \left(\overline{R} + \sum_{m=1}^{j-1} \binom{i}{m} Q^m \cdot \overline{Q}^{i-m} \right) \end{aligned} \tag{16}$$

Table IV shows the values of S_j with $j = \{1, \ldots, 8\}$, while table III shows the intermediate results of T_j. The final column in table III also lists the expected values, i.e., the expected number of tags that have an overlap of x bit-positions per readout (for $x = \{1, \ldots, 8\}$).

From these results, we can for example deduct that on average, only 3.298 tags have a bit-overlap of 3 or more bits when two 10-item sets are read (see row "x=3", column "expected value" in table III). Table IV shows that on average, every second readout will find at least 1 tag with a bit-position overlap of 4, but finding 4 or more tags with a bit-position overlap of 4 will only occur at every 800^{th} readout. Finding anything more than 7 overlapping bit-positions is very unlikely, even when requiring only 1 item to have such an overlap (see row "x=7", column ">=1" in table IV). In most cases, 7 or 8 tags will have an overlap of 2 bit positions with a previous readout (see row "x=2", column ">=7" in table IV), but already finding an overlap of 3 bit-position is much less likely: typically no more than 5 tags have that many overlapping bit-positions (see row "x=3", where columns ">=0" through ">=5" represent over 90% of all cases).

Coming back to our original question of how well Shamir Tags could be tracked in practice (section III-C), we can now turn to table II to see the effects of typical bit-overlaps. For example, finding an overlap at as many as 4 bit-positions,[11] given a population of several million tags, more than half a million items would share those same 4 bit values. Consequently, if a reader repeatedly encounters those 4 bit-positions with identical values, tracking is difficult. Even when up to 7 bit-positions overlap, which is an extremely unlikely case, this only yields a resolution of 10 000 items in a (small) population of 1 million.

[11]Cf. row "x=4" in table IV: Finding 4 overlapping bit-positions happens in less than 50% of all cases, and for 1 or 2 tags only.

IV. Discussion

We now discuss practical aspects of designing a privacy protection system based on Shamir Tags. In particular, we need to find suitable values for the number and size of the Shamir shares, as well as the bit disclosure parameters. We will also need to think about practical attacks and deployment costs.

A. System Parameters

We based our analysis above on a Shamir Tag with 3 shares, totalling 318 bits and disclosing 16 bits upon first read (cf. Fig. 3). Obviously, other choices are possible.

Fig. 5 gives an overview of the readout process and its corresponding system parameters. After a reader has powered a Shamir Tag at time t_B, an initial set of bits is disclosed in bulk. The size of this set should be such that a tag can instantly be identified from a known (i.e., cached) tag population. We call this number CIL – the *cached identification limit*. After an initial delay of T_D, the tag then releases individual bits (cf. Fig. 3) at a constant disclosure rate δ. If the tag leaves the reader vicinity at time t_E (i.e., before all bits have been disclosed), the thus obtained random bit-substring does not allow the reader to infer the item's true ID. Only if the reader powers the tag for the entire interval $T_F = T_D + (\mathrm{TotalBits} - \mathrm{CIL})/\delta$ can the tag's true ID be computed from scratch.[12]

The size of CIL depends on the envisioned cached population, e.g., the size of a personal household, or the number of items arriving daily at an industrial loading dock. It should be large enough so that the particular amount of bits is able to differentiate all items currently in the cache. Table II already gave an overview of possible choices for CIL. While private citizens might not have more than a few

[12]This of course assumes that no cached information is available – otherwise one can guess the remaining bits based on the known tag population (cf. section III-A).

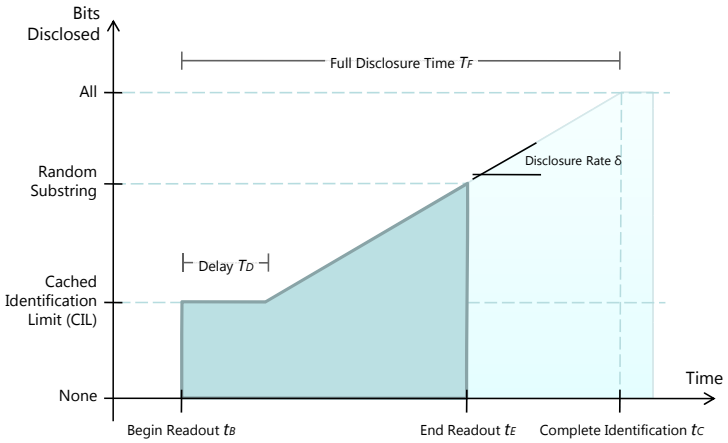

Fig. 5. *System Parameters.* After disclosing an initial number of CIL bits for cached identification, bits are randomly disclosed at a rate of δ bits per seconds after a delay of T_D seconds. Only if the readout process ends at or after t_c has been reached (i.e., after T_F seconds), the full ID can be reconstructed.

thousand tagged items, it might be desirable to plan for larger populations, in particular for being able to support industrial scenarios. With our choice of 16 bits, some 50 000 cached items can be reasonably identified. Using as many as 20 bits would support a reliable identification among 1 million cached items.

After having disclosed CIL bits, the remaining number of bits on a Shamir Tag should be large enough to make brute-force attacks infeasible. Obviously, using more Shamir shares on each tag will help, yet only until the tags begin the bitwise release after the time interval T_D. Assuming a target disclosure interval T_F, increasing the number of Shamir shares requires increasing the disclosure rate δ accordingly. This means that during the interval $T_F - T_D$, the effect of a larger number of shares is nullified. Our initial choice of only 3 Shamir shares per tag leaves enough bits after the initial disclosure (about 300 bits) to render brute-force infeasible, while requiring only 318 bit total tag payload.

Also during the initial interval T_D, tracking resistance is improved by using a larger number of shares on a tag, as fewer bit positions will overlap between individual readouts. Again, after interval T_D, the steeper rate of δ (assuming that we want to keep T_F constant) will gradually diminish this advantage.

Given these observations, the main system design parameters will be the size of CIL and the total number of Shamir shares on each tag, as well as the envisioned full disclosure time interval T_F and the initial delay T_D. Reasonable values for T_D would be several seconds, maybe up to 10, which would eliminate the majority of all unauthorized gate disclosures (e.g., readers in doors), as people would pass through in a few seconds and thus only disclose the initial CIL bits. For the overall time needed to read out a complete tag, T_F might be as much as several minutes, which would still allow storage applications and customer returns, but would require an unauthorized reader to follow a mobile user for prolonged time in order to continuously power the tag.

Fig. 6 shows the relationship between the size of CIL and the number of Shamir shares stored on the tag. It uses the *cumulative overlap value* (COV) of a 10-tag readout with another 10-tag readout to illustrate the traceability protection levels of different parameter choices. COV is computed directly from the last column ("expected value") in table III by summing up the individual expected position-overlap values $E(x)$:

$$\mathrm{COV} = E(20) \cdot 20 + \sum_{i=19}^{0} \left(E(i) - \sum_{j=i+1}^{20} E(j) \right) \cdot i \tag{17}$$

Our initial choice of 3 Shamir shares and a CIL of 16 bit has a COV of 22.65 bits – using 4 shares would bring it down to 19.01 bits. Note that the value of COV is simply for comparison purposes, it does not mean that there is a position overlap of 22 bits when comparing two readouts! However, it illustrates that, for a given CIL, one should aim for a larger number of shares in order to lower the chances for bit overlaps from disclosing the initial

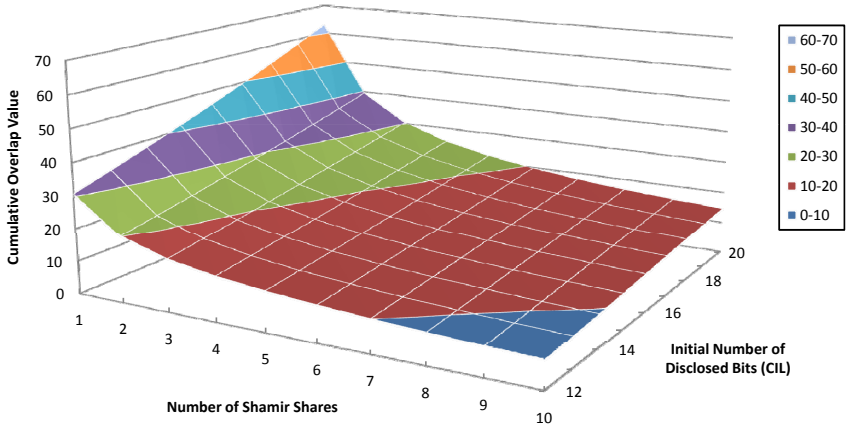

Fig. 6. *Illustrating the Tradeoff between Initial Bit Disclosure (CIL) and Number of Shamir Shares Using the Cumulative Overlap Value (COV).* By summing up the individual expected overlap values in table III for different choices of CIL and number of Shamir shares, we can illustrate the traceability of a certain choice of parameters. Lower values of COV are better.

set of bits. For example, to provide the same level of protection when increasing CIL from 16 to 20 bits, we can see in Fig. 6 that we should use at least 5 shares (which has a COV of 21.98).

B. Attacks

While our analysis has shown that each readout will only share a few bit-positions with other readouts, a careful study of a reader's log file might still reveal patterns and trends that could support the tracking of tagged items. Below we list some options for analyzing such overlaps, though an exhaustive treatment of the data mining aspects of such information is beyond the scope of this paper. Perhaps the most important aspect of any such attack is that it basically remains opportunistic, i.e., it might be able to uncover partial tracks of selected items, yet this will very much depend on the law of large numbers and pure chance.

We would like to stress our initial goals again: to construct a system that will prevent *casual* and *accidental* (i.e., unwanted) readouts – not to offer 100%-percent protection from identification and traceability. If an attacker is determined to track a known target over the course of days and months, doing so through someone's shopping items or clothing might very well be possible, even if they are protected by Shamir Tags. However, there should in any case be easier and more reliable methods at such an attacker's disposal (e.g., tracking devices or cell phone logs). Having said that, we still need to point out our system's weaknesses:

a) Windowing: A windowing attack utilizes the fact that out of the total tag population, the spatial distribution of tags is highly variable. For example, while a reader installed in the main train station of a city might see not only the tags of the local citizens, but also those from visitors and tourists, a reader installed in a small neighborhood supermarket might see much fewer tags over the course of days and month. Putting a reader in front of a small office building will most likely encounter only a few thousand tags from less than 100 people, few enough so that tracking is possible even with position-overlaps of 5 or 6 bits. Table II shows that with a 6 bit overlap, a small population of several hundred tags could potentially be tracked.

b) Merging: A merging attack tries to reconcile multiple substring readouts into larger fragments, very much like a crossword or sudoku puzzle. In our example in Fig. 4, the overlapping bits of readout 2 could be used to construct a more complete version of the original readout from tag 1. Note, however, that an attacker would also need to take into account alternative hypotheses, as the original readout could have equally likely been from tag 2. Consequently, the original readout would need to be combined with the data from both readout 1 and readout 2 to create two hypothetical tags, of which only one (the combination with readout 2) would actually represent an existing tag. Combining the original readout with readout 1 will create a (most likely) non-existent tag, as it combines bits from two different tags (tag 1 and tag 2). With

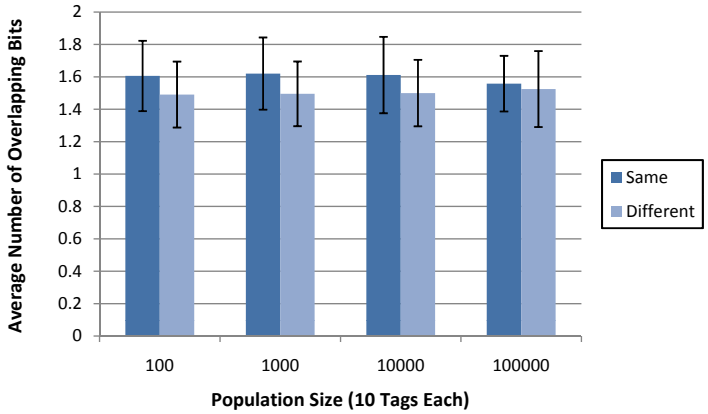

Fig. 7. *Average Number of Overlapping Bit-Values for Tag-Sets from Identical and Different Persons (with Error-Bars).* Simulating different user populations of $n = \{100, 1000, 10\,000, 100\,000\}$ carrying 10 tags each, we performed 2000 readouts and computed the pairwise number of overlapping bit-*values* (i.e., those where the overlapping bit-positions had identical values). Tag readouts from the same person tend to feature more overlapping bit-values than those from different people. However, the large standard deviations around those means make it difficult to infer "sameness" from such an average figure.

thousands of tags sharing the same 5 or 6 bit positions, the repeated combination of overlapping bits will quickly create a combinatorial explosion, as n overlapping tags result in $\binom{n}{2}$ possible combinations. For example, finding some 10 000 overlapping bit-strings in a reader's log, one could construct 50 million possible combinations.

c) Statistical Analysis: While the number of overlapping bit positions (cf. tables III and IV) is irrespective of the similarity of the encountered tags, the actual number of overlapping *bit-values* across all carried tags might be larger if one encounters a previously read-out set of tags. This could allow an attacker to infer whether the just-read set of bits is from a previously encountered set, thus aiding with a merging attack. Again, combinatorics quickly makes such an attack expensive, as a pairwise comparison requires $\binom{n}{2}$ operations (just as in a merging attack). In order to get a feel for the discriminatory power of such a heuristic, we simulated different populations sizes of 100, 1000, 10 000 and 100 000 people, each carrying 10 tags. In each experiment, we performed 2000 random readouts from the population. We then conducted a pairwise comparison among those 2000 readouts, resulting in $\binom{2000}{2} = 1\,999\,000$ comparisons. For each comparison, we noted whether the two readouts were from the same person or a different one. A comparison implied matching each of the 10 bit-readouts with all 10 other bit-masks from the other readout, resulting in 100 bit-mask comparisons. Note that we explicitly did *not* discard comparisons where one or more bits did have different values. While these are clearly from different tags, we cannot discard the entire match, as a person might have simply exchanged one item for another. We instead computed an *average bits matched* as follows: For each of the 10 bit-masks from the first readout, the maximum bit overlap with any of the other readouts' bit-masks is determined. These 10 maximums are normalized and used as an overall overlap characteristic for the readout pair. Fig. 7 summarizes the mean and standard deviation for the four different user bases we performed this experiment on. Table V summarizes the results from the corresponding t-tests. The differences are almost always statistically significant ($\alpha < 0.05$, see last row), except for the largest population, where the 2000 readouts produced only very few reads from identical persons.[13]

C. Deployment

In contrast to most of today's proposed RFID privacy technology [5], deploying Shamir Tags is simple and cost-efficient. This is due to two reasons:

[13]In order to perform t-test comparisons on our data, we needed to have the same number of observations in both data sets. To do so, we randomly selected as many results from the "different"-set as we had results in the "same"-set (since we typically got many fewer results from the "same" person). Randomized selection was performed on the raw text file containing one result per line, using the free line randomization tool `rl` from Arthur de Jong (see `ch.tudelft.nl/~arthur/rl/`). In total, we always had $\binom{2000}{2} = 1\,999\,000$ observations, so one can compute the total number of "different"-observations that we got by subtracting the number of "same"-observations in table V from 1 999 000.

TABLE V

t-TESTS FOR SIMULATION RESULTS USING DIFFERENT POPULATION SIZES. FOR COMPARISON PURPOSES, THE RESULTS FOR THE "DIFFERENT"-SETS ARE RANDOMLY CHOSEN SUBSETS TO MATCH THE SIZE OF THE CORRESPONDING "SAME"-SET.

Population (People)	100		1000		10000		100000	
Bit-Overlap (Values) w/	*Same*	*Different*	*Same*	*Different*	*Same*	*Different*	*Same*	*Different*
Mean	1.60580	1.49087	1.62039	1.49541	1.61146	1.50000	1.55833	1.52500
Variance	0.04715	0.04140	0.04967	0.03990	0.05557	0.04209	0.02949	0.05500
Observations	19817	19817	2070	2070	192	192	24	24
t Stat	5.4240E+01		1.9433E+01		4.7194E+00		5.4654E-01	
P(T<=t) one-tail	0.0000E+00		1.0652E-77		2.2821E-06		2.9498E-01	

1) *No cryptographic functionality on the tag or in the reader:* Shamir Tags simply store a (large) number. Any needed cryptographic operations are either employed during tag creation (at the manufacturer's site), or in the application reading the tag (for recombining the Shamir shares). Neither the reader nor the tags will need any modification from today's standards, except for bit-throttling support. The required random number generator (RNG) for creating temporary session IDs (cf. section II-A) is already part of today's tag standards [3].
2) *Moderate size requirements:* Shamir Tags can work with as few as 40 bytes of tag memory. Today's passive RFID tags already feature several hundred bits of memory, some more than 1000 bits [20]. While using a 318-bit Shamir share for storing a 96-bit EPC will increase costs over storing the data in plaintext on the tag, the added costs are minimal, as today's majority of costs for manufacturing an RFID-tag comes from assembling chip and antenna, not from the chip's silicon [21].

Due to our initial design goal of abolishing the need for passwords and lookup-lists, no added point-of-sales requirements would be introduced, i.e., consumers would not require specific privacy-enabling equipment (e.g., a "guardian" device, such as a special mobile phone) to enjoy a basic level of protection right after purchase.

V. CONCLUSIONS

The analysis above has shown that our approach is able to provide instant identification of *personal* (i.e., known) items, using an initially disclosed set of bits. At the same time, it is able to provide protection from unauthorized identification, as items can only be identified if *all* bits on the tag have been disclosed, which requires a significant amount of reading time due to the employed bit-throttling mechanism. If, however, all bits have been read, the original ID of the tagged item can easily be computed, alleviating the need for cumbersome (and costly) password management. Last but not least, our method is also effective against unauthorized tracking, since the random selection of individual bits leads only to small overlaps, which will in most cases not be enough to repeatedly track an item with any reasonable confidence.

Together with corresponding legislation that makes the unauthorized reading of tags a crime, citizens could rest assured that unscrupulous retailers cannot claim accidental readouts, as the initial CIL would allow them to immediately identify their own goods (i.e., those not already sold). Storing sets of bits from unknown or already sold merchandise would be in clear violation of existing laws and could thus be persecuted, given a corresponding auditing process. Clearly, our solution does not protect against persistent attackers who devote much time and energy to secretly follow mobile users or install hidden reading devices in places where individuals rest for extended periods of time (e.g., public transportation). However, given our aim of providing basic protection for cheap and abundant items such as clothing or groceries, such an attacker might favor more efficient surveillance capabilities in the first place.

ACKNOWLEDGEMENTS

F. Mattern, Ch. Roduner, and S. Santini provided valuable feedback and advice on earlier versions of this paper.

REFERENCES

[1] K. Albrecht, "Supermarket cards: The tip of the retail surveillance iceberg," *Denver University Law Review*, vol. 79,

no. 4, pp. 534–539, 558–565, Oct. 2002. [Online]. Available: www.nocards.org/AutoID/overview.shtml

[2] A. Kantor, "Tiny transmitters give retailers, privacy advocates goosebumps," USAToday.com – CyberSpeak, December 19, 2003.

[3] EPCglobal, "Class-1 generation-2 UHF RFID protocol for communications at 860 MHz–960 MHz, version 1.0.9, EPC Radio-Frequency Identity Protocols," Jan. 2005. [Online]. Available: www.epcglobalinc.org/standards/Class_1_Generation_2_UHF_Air_Interface_Protocol_Standard_Version_1.0.9.pdf

[4] G. Karjoth and P. A. Moskowitz, "Disabling RFID tags with visible confirmation: clipped tags are silenced." in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES 2005)*, V. Atluri, S. De Capitani di Vimercati, and R. Dingledine, Eds. Alexandria, VA, USA: ACM Press, 2005, pp. 27–30.

[5] A. Juels, "RFID security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, Feb. 2006. [Online]. Available: www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/pdfs/rfid_survey_28_09_05.pdf

[6] A. Juels, D. Molnar, and D. Wagner, "Security and privacy issues in e-passports," in *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. Washington, DC, USA: IEEE Computer Society, Sept. 2005, pp. 74–88. [Online]. Available: http://www.cs.berkeley.edu/~daw/papers/epassports-sc05.pdf

[7] S. A. Weis, "Security and privacy in radio-frequency identification devices," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2003. [Online]. Available: theory.lcs.mit.edu/~sweis/

[8] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," in *Security in Pervasive Computing – First International Conference, Boppard, Germany, March 12–14, 2003, Revised Papers*, ser. LNCS, D. Hutter, G. Müller, W. Stephan, and M. Ullmann, Eds., vol. 2802. Berlin Heidelberg New York: Springer, 2003, pp. 201–212. [Online]. Available: www.springerlink.com/openurl.asp?genre=issue&issn=0302-9743&volume=2802

[9] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic approach to "privacy-friendly" tags," in *RFID: Applications, Security, and Privacy*, S. Garfinkel and B. Rosenberg, Eds. Addison-Wesley, July 2005. [Online]. Available: www.rfidprivacy.us/2003/papers/ohkubo.pdf

[10] D. Henrici and P. Müller, "Tackling security and privacy issues in radio frequency identification devices," in *Pervasive Computing – Second International Conference, PERVASIVE 2004, Vienna Austria, April 21–23, 2004, Proceedings*, ser. LNCS, A. Ferscha and F. Mattern, Eds., vol. 3001. Berlin Heidelberg New York: Springer, Apr. 2004, pp. 219–224. [Online]. Available: www.springerlink.com/openurl.asp?genre=issue&issn=0302-9743&volume=3001

[11] A. Juels, "Minimalist cryptography for RFID tags," in *Security of Communication Networks (SCN)*, C. Blundo, Ed., Amalfi, Italy, Sept. 2004. [Online]. Available: http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/minimalist/Minimalist.pdf

[12] R. Stapleton-Gray, "Scanning the horizon: A skeptical view of RFIDs on the shelves," July 2005. [Online]. Available: www.rfidprivacy.us/2003/papers/stapleton-gray3.pdf

[13] A. Juels, R. L. Rivest, and M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy," in *Proceedings of the 10th ACM Conference on Computer and Communication Security*, S. Jajodia, V. Atluri, and T. Jaeger, Eds. Washington, D.C., USA: ACM Press, 2003, pp. 103–111. [Online]. Available: portal.acm.org/citation.cfm?id=948126&coll=Portal

[14] K. Fishkin, S. Roy, and B. Jiang, "Some methods for privacy in RFID communication," in *Security in Ad-hoc and Sensor Networks – First European Workshop, ESAS 2004, Heidelberg, Germany, August 6, 2004, Revised Selected Papers*, ser. LNCS, C. Castelluccia, H. Hartenstein, C. Paar, and D. Westhoff, Eds., vol. 3313. Berlin Heidelberg New York: Springer, August 2005, pp. 42–53.

[15] A. Shamir, "How to share a secret," *Comm. of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] EPCglobal, "EPC tag data specification 1.3, EPCglobal Standard," Mar. 2006. [Online]. Available: www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf

[17] NFC Forum, "Near Field Communication (NFC)," www.nfc-forum.org. [Online]. Available: www.nfc-forum.org

[18] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. Boca Raton, Florida: CRC Press, 1996. [Online]. Available: http://www.cacr.math.uwaterloo.ca/hac

[19] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[20] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. Wiley, 2003.

[21] G. Swamy and S. Sarma, "Manufacturing cost simulations for low cost RFID systems," Auto-ID Center," White Paper, 2003. [Online]. Available: www.autoidlabs.org/uploads/media/MIT-AUTOID-WH017.pdf

[22] S. Garfinkel and B. Rosenberg, Eds., *RFID: Applications, Security, and Privacy*. Addison-Wesley, July 2005. [Online]. Available: www.rfidprivacy.us

Marc Langheinrich is a senior researcher in the Institute for Pervasive Computing at the ETH Zurich, Switzerland. He received a Master's degree (Dipl.-Inf.) in computer science from the University of Bielefeld, Germany, in 1997, and a PhD (Dr. sc.) in the areas of ubiquitous computing and privacy from the ETH Zurich, Switzerland, in 2005.

Before joining the ETH Zurich in 1999, he worked for several years as a researcher in both academia (Univ. of Washington, Seattle) and industry (NEC Research, Tokyo). Marc is one of the authors of P3P, a W3C-standard for privacy on the Web, and has published extensively on privacy aspects of ubiquitous and pervasive computing systems.

Remo Marti is a software engineer with Ergon Informatik AG, Zurich, Switzerland. He received a Master's degree in computer science from the ETH Zurich in 2006.