Physically Unclonable Function -Based Security and Privacy in RFID Systems *

Leonid Bolotnyy and Gabriel Robins
Department of Computer Science, University of Virginia
{lb9xk, robins}@cs.virginia.edu

Abstract

Radio Frequency Identification (RFID) is an increasingly popular technology that uses radio signals for object identification. Tracking and authentication in RFID tags have raised many privacy and security concerns. On the other hand, known privacy and security cryptographic defenses are too hardware-expensive to incorporate into low-cost RFID tags. In this paper, we propose hardware-based approaches to RFID security that rely on physically unclonable functions (PUFs). These functions exploit the inherent variability of wire delays and parasitic gate delays in manufactured circuits, and may be implemented with an order-of-magnitude reduction in gate count as compared with traditional cryptographic functions. We describe protocols for privacy-preserving tag identification and secure message authentication codes. We compare PUFs to digital cryptographic functions, address other uses of PUFs to enhance RFID security and suggest interesting directions for future research. The proposed solutions are efficient, practical, and appropriate for low-cost RFID systems.

1 Introduction

Existing Radio Frequency Identification (RFID) security algorithms rely on digital cryptographic primitives that require relatively high hardware cost (e.g., thousands of gates per tag to implement [6], which could be prohibitive for cheap RFID tags). We propose a low-cost hardware-based approach to RFID security that requires only hundreds of gates to implement, and is based on physically unclonable functions (PUFs).

A *physically unclonable function* (PUF) is a random number function that can only be evaluated by a specific instance of the underlying hardware [8]. A PUF computes its output by exploiting the inherent variability of wire delays and gate delays in manufactured circuits. These delays in turn depend on highly unpredictable factors, such as manufacturing variations, quantum mechanical fluctuations, thermal gradients, electromigration effects, parasitics, noise, etc. A good PUF is therefore not likely to be accurately modeled succinctly, nor be predicted or replicated, even using identical hardware (which will still have different random manufacturing variations and associated delays, and thus yield an implemented function different from the first). Moreover, PUFs may be implemented with an order-of-magnitude reduction in gate count, as compared with traditional cryptographic functions.

*This research was supported by a Packard Foundation Fellowship, by NSF Young Investigator Award MIP-9457412, and by NSF grants CCR-9988331 and CCF-0429737. For additional related papers see http://www.cs.virginia.edu/robins

The idea of a physically unclonable function originated in [16] with an emphasis on optical PUFs. A silicon PUF prototype was designed in [7] [12] based on relative delay comparisons. Their design is highly reliable, can tolerate varying environmental conditions, and contains enough manufacturing variations to make each PUF circuit substantially different from the same PUFs in other chips. Although not proven, an important characteristic of the PUF proposed in [12] is that it is difficult to create an accurate model for it based on at most polynomially-many known input/output pairs.

In [17], the authors propose an off-line reader authentication algorithm based on PUFs that uses public key cryptography, which is still prohibitively expensive for low-cost RFID tags. A simple PUF-based identification/authentication algorithm for RFID, based on [8], was proposed in [15]. In this scheme the back-end system learns many challenge-response pairs for each PUF circuit (i.e., each tag), and then uses hundreds of challenges at a time to identify and authenticate millions of tags, probabilistically ensuring unique identification [8] [15]. However, the lack of access control provisions in this approach exposes tags to identification by adversarial readers. Moreover, tags do not maintain a state and do not use any randomness in their responses, making them vulnerable to tracking. Since numerous challenges are necessary for single tag identification, many communication rounds between the reader and the tag are necessary, increasing the required identification time and the tag's power consumption. In light of these shortcomings, we argue that it is more appropriate to use a challenge-response communication scheme for authentication purposes only. Sending a tag ID to the reader first will allow the reader to greatly reduce the number of communication rounds with a tag and to query tags with different challenges; still, tag tracking remains possible.

There are two important PUF parameters pivotal to our discussion. The first parameter, denoted τ, represents the probability that the PUF output for a random input challenge is the same as the PUF output of another identical tag for the same challenge. To empirically compute τ for n tags, we can determine how many of the $\binom{n}{2}$ possible tag pairs produce identical outputs for a given challenge. The probability τ is computed as the ratio of such pairs to the total number of pairs. The second parameter, denoted μ, represents the probability that the output of a PUF for a given challenge is different from the PUF's reference output (i.e., the output recorded in a stable controlled reference environment). Different operational conditions can affect μ, e.g., variations in temperature, voltage, and circuit manufacturing variability [12]. All of these factors need to be considered when designing a PUF-based RFID system.

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

2 PUF Assumptions

We assume that an adversary cannot construct an accurate model of a PUF from a polynomial number of known input/output pairs (i.e., polynomial in the PUF's size or its input/output bit-lengths). We also assume that the pairwise PUF collision probability τ is constant for any challenge, independent of the number of identical responses that tags output to other challenges. These assumptions are justifiable based on the practical experiments discussed in [12]. We postulate that physical tampering with a tag's PUF or probing it to measure actual wire delays will modify the PUF and thereby destroy the very function that it is designed to compute. We assume that the PUF is secure against side-channel attacks that try to recover the key (i.e., the PUF's computational behavior). Relying on these assumptions, we will sketch proofs of the privacy/security of our algorithms, considering only algorithmic attacks and assuming that a PUF is a random function.

3 PUF-Based Tag Identification Algorithm

We propose a simple, single-use 1-step identification algorithm for preserving privacy against passive adversaries. An algorithm is privacy preserving if an adversary can not distinguish between any pair of tags. We will elucidate the underlying assumptions after we describe the algorithm, since they will be more readily understood in the context of the algorithm. Our tag identification algorithm is based on the classical cryptographic idea of using pseudonyms or one-time-pads to provide security.

3.1 Related Work

Several previous works use pseudonyms to provide RFID privacy. For example, the "minimalist-approach" of [9] uses legitimate readers to update pseudonyms aboard a tag after each authentication. Using PUFs to generate new pseudonyms allows pseudonyms to be updated much less frequently. Another scheme builds hash-chains [1] and requires tags to implement two relatively costly digital cryptographic hash functions. In contrast, our scheme only needs a single PUF (since the function p is secret / random / unpredictable to an adversary), while still remaining unsusceptible to simple tag tracking. If the PUFs are one-way functions (but not necessarily collision resistant), we can use PUFs in place of the two hash functions of [1] to protect secret tag identifiers.

3.2 Our Tag Identification Algorithm

We now describe our PUF-based tag identification protocol. Let ID be an identifier stored aboard a tag, and let p denote the tag's PUF. When a reader interrogates a tag, the tag responds with ID and updates its identifier to $p(\mathrm{ID})$. The reader looks up ID in its database to determine the tag's identity. This implies that the back-end database needs to store the sequence $\mathrm{ID}, p(\mathrm{ID}), p^{(2)}(\mathrm{ID}), \ldots, p^{(k)}(\mathrm{ID})$ for each tag, where $p^{(i)}$ denotes the composition of the PUF p with itself i times.

Note that it is important for the PUF responses to be reliable (i.e., return consistently equal output responses for the same inputs), since otherwise errors will compound in long chains of PUF compositions. To address this issue, we propose running p multiple times for the same ID at each stage of the composition, selecting the majority answer for the new ID, to ensure that the value of the new ID is stable and reliable (i.e., unperturbed by random environmental factors). For example, in a reference environment where the probability of an unreliable PUF value is $\mu = 0.02$, if we execute the PUF $N = 5$ times at each PUF invocation, the reliability of the last value of $k = 100$ PUF compositions can be estimated as $R(\mu, N, k) \geq (1 - \sum_{m=\frac{N+1}{2}}^{N} \binom{N}{m} \mu^m (1-\mu)^{N-m})^k \approx 0.992268$. Thus the reliability of a large number of PUF compositions can be made arbitrarily high using only a modest number of repetitions at each stage of the composition chain. Note that without such iteration at each stage of the composition, the expected length of a reliable composition chain is smaller (e.g., for $\mu = 2\%$ and no iterations, i.e., $N = 1$ at each stage, the expected chain length before an erroneous value appears is $\frac{1-\mu}{\mu} = \frac{1-0.02}{0.02} = 49$).

Another technique for increasing the probability of tag identification using PUF-computed IDs is to run several PUFs in parallel, each independently generating its own composition chain. At each query, the tag sends a *tuple* of PUF values to the reader (i.e., one value for every PUF aboard the tag). The reader will continue normal identification operations as long as at least one of the PUF chains is still valid (i.e., a tuple of PUF compositions is considered to be reliable if at least one of its component PUF values is reliable). Since the variance of a geometric distribution is high, such a "super chain" of tuples is expected to remain valid longer than any of its individual component PUF chains. In particular, for a tuple of size q (i.e., using q independent composition chains in parallel), the expected number of consecutive successful identifications is:

$$S = \sum_{x=1}^{\infty} x \cdot [(1 - (1-\mu)^{x+1})^q - (1 - (1-\mu)^x)^q]$$

For example, for a tuple of size $q = 2$, we have $S \approx 73$, and for a tuple of size $q = 3$, the expected length of a valid chain rises to $S \approx 90$. This multiple-chain strategy increases the overall probability of successful tag identification. Having several PUFs aboard a tag is not an unreasonable hardware burden, since PUFs can be implemented using only a small number of gates. Alternatively, a single PUF can be used to simulate multiple PUFs by using an extra input PUF parameter to select from a family of different PUF functionalities. We can combine the two techniques by iterating each tuple component multiple times, thus increasing the overall reliability.

In all of the above strategies, once an expected number of reliable pseudonyms is exhausted, a tag can start a new chain using a new seed identifier. Such seed identifiers can be stored aboard the tag, or provided to the tag by the reader (which implies that the back-end database must store several PUF value composition chains for each tag). In summary, given a reasonable strategy to maintain reliability, a good PUF can extract from a single seed identifier many pseudonyms that can be used to privately identify tags.

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

3.3 Assumptions and Requirements

We assume that the adversary does not carry out a denial of service attack (e.g., this can be enforced by considering only passive adversaries). We assume that an adversary cannot physically overwrite identifiers on tags that it may own without damaging (or at least altering) their PUF circuits. This assumption is important for the PUF of [12] that we rely upon, as it has a relatively large tag differentiation τ value (e.g., $\tau = 0.4$ [12]). If a better PUF is available (i.e., one with a smaller τ value), this assumption may become unnecessary.

We place the following requirements on secret key construction by the back-end system. For the algorithm to ensure tag privacy, the PUF must be able to generate long chains of unique IDs (i.e., without repetitions). Moreover, different tags should not yield identical PUF outputs. This implies that the number of possible inputs/outputs should be significantly larger than the number of tags in the system. If an ID does repeat, the algorithm chooses a different ID for that tag. Since tags update their internal state after each read, the reader must supply enough power to the tags to support write operations (as write operations require more power than reads).

3.4 Adversarial Model

The privacy of the identification algorithm will be determined from an experiment performed by an adversary. An adversary will observe the reader's communication with multiple tags (at most polynomially-many rounds in the bit-length of PUF input), and single out two tags. The reader will then randomly select one of these two tags and run the identification algorithm once. An adversary is successful in compromising a tag's privacy if it can determine which of the two tags the reader has selected with a probability substantially greater than $\frac{1}{2}$ (i.e., better than simply guessing).

Theorem 3.1 *Given a random oracle assumption [2] for PUFs, an adversary has no advantage in attempting to compromise a tag's privacy.*

Proof sketch: By observing the non-repeating output sequences of any two tags and receiving the next output from one of them, an adversary cannot determine which one of the two PUF functions computed it, since PUFs are assumed to behave as random functions. □

4 PUF-Based MAC Protocols

A message authentication code (MAC) protocol is a three-tuple (K, T, V) where K is the key generation algorithm, T is the tagging algorithm, and V is the verification algorithm. The algorithm K generates a key for use by algorithms T and V. The tagging algorithm takes message m as the input, and outputs its signature σ. The verification algorithm verifies that the signature σ for a message m is authentic. A MAC protocol is secure if it is resistant against forgeries. An adversary is successful in forging a signature if it can create a valid signature for a message whose signature it has not seen before.

4.1 Related Work

A message authentication code (MAC) aboard a tag can be implemented using a standard cryptographic hash function (e.g, MD5, SHA-256), or be based on a block cipher such as the Advanced Encryption Standard (AES). Alternatively, for low-cost implementation, a one-time signature scheme [11] can be used, as noted in [10], where each bit position of the signature has two associated secrets - one for 0 and one for 1. Then, the signature is an ordered sequence of secrets that corresponds to 0 or 1 in each bit position. However, such a scheme requires a prohibitive amount of memory aboard a tag. A more "minimalistic" approach, where each secret is only a single bit, is suggested in [10], which, in order to avoid simple forgeries, lengthens the message size and makes the message space sparser. This scheme allows the construction of a one-time MAC. We suggest a different PUF-based MAC implementation that is efficient and allows messages to be signed multiple times.

4.2 Our MAC Protocols

Our PUF-based MAC algorithms use multiple PUF computations to sign a message. Note that in a tag authentication algorithm the reader authenticates a tag, whereas in a MAC algorithm, a tag signs / authenticates a message. The message the tag signs is used as the input to a PUF, and the PUF's output is the signature / MAC. We give two MAC protocols that rely on a PUF for security. The choice of protocol depends on the size of the message space.

Before presenting the algorithms, we emphasize that the key used to sign a message is the PUF itself. Consequently, if the message space is small, the back-end database can make the required precomputations to learn the desired behavior of the PUF. However, if the message space is large, such exhaustive precomputations are infeasible. Therefore, the size of the message space dictates the tag design requirements and drastically affects the preferred MAC protocols used.

The PUF-based MAC protocols we describe are different from standard cryptographic MAC algorithms. First, the keys that the verifier requires to validate a signature are large, whereas standard cryptographic MACs have short keys. Second, one of our MAC algorithms cannot verify a signature without the physical presence of the tag that signed the message, and our other algorithm can not sign arbitrary messages. These properties are unusual for classic cryptographic algorithms based on digital secrets, but are appropriate for resource-constrained RFID systems. In order to keep tag cost down, the computational burden is pushed to the back-end system.

Our protocols cannot be applied to all scenarios requiring MAC computations aboard tags (e.g., where the message space is large and MAC verification must be performed when a tag is not within range of a verifier). However, PUFs can still be used for general MAC constructions as part of a key generation algorithm, thus preventing physical attacks on otherwise vulnerable tags. Our MAC protocols can be used in some applications of "yoking-proofs" [4] [10], where confirmation is sought that a group of tags are read simultaneously.

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

Our proposed MAC protocols assume a powerful adversary that can adaptively select up to polynomially-many (m, σ) pairs from which it seeks to forge valid future signatures. We designed our protocols to resist forgery even if the signature verifiers are off-line (i.e., they do not participate in the signing algorithm), as is required in [4] [10]. We will state the scheme-specific assumptions below, during the discussions of individual protocols.

Before presenting the algorithms, we give an example of where our MAC constructions can be used. In this example, heat sensitive objects are tagged with RFID tags containing unpowered temperature sensors [14]. As these perishable items transit from the seller/supplier to the buyer/consumer, their temperature must provably not exit a specified range. The buyer's readers in the vicinity of the objects will collect temperature readings from the tags during transport. To provide the temperature range guarantee, the buyer's readers will request that the sensing RFID tags sign the temperature values. When the shipment arrives at its destination, the buyer will verify that the temperatures/messages signed by the objects are authentic. This example is applicable to the case when the size of the message space is large and signatures need to be verified in the vicinity of the tag that signed the message. This example is also valid when the size of the message space is small (e.g., if the temperature values are discretized or rounded to the nearest degree).

4.3 Large Message Spaces

When the size of the message space is large, we cannot perform all of the desired PUF computations before deploying the tags. Consequently, signature verification can only be performed when a tag in transit is within reader range (e.g., when it arrives at its destination). For now we disallow hardware tampering attacks; later we will discuss how the issue of hardware tampering can be addressed.

The essential basis of a MAC signature for PUF p and message m is $p(m)$. To prevent unauthorized reuse (or replay) of such a signature, we modify the behavior of the PUF p using a unique token c to yield a more general parametrized PUF function $p_c(m)$ whose behavior depends on c. Since a passive tag does not have a clock aboard, it can instead prevent a replay attack by using a random number or a counter. Using a counter as a unique token instead of a random number has the advantage that it timestamps the message and creates a natural total ordering of the signatures (e.g., in our buyer/seller example above, the replay of past signatures will be easily detected with a counter-based scheme). However, it also has the disadvantage of revealing the state of a tag, which could leak private information aboard the tag. Note that randomness can be used in PUF computations as well, since signatures are verified in the presence of their generating tag. The signature algorithm does not accept input from the reader to allow for off-line verification. In on-line scenarios, readers can supply inputs to the signing PUF.

To reduce the probability τ of forgery using the computations of other tags, multiple PUF computations are required to create a signature. Therefore, the signature for a message m is $\{c, r_1, \ldots, r_n, p_c(r_1, m), \ldots, p_c(r_n, m)\}$, where $r_1, \ldots, r_n$ are different random numbers generated aboard the tag. This computation is similar to the one needed for tag authentication. However, in the MAC algorithm, the random input to the PUF is generated by the tag rather than the reader, and it includes a timestamp c.

The choice of n (the number of PUF computations) depends on τ, μ (defined above), and the application requirements. To quantify the reliability of message authentication and the difficulty for an adversary to forge a signature using equivalent PUFs, we compute the probability prob_v that a *valid* signature is verified as authentic, and also the probability prob_f that a *forged* signature is incorrectly determined to be valid by the verifier. Because μ in [12] is non-negligible, some error(s) may be allowed in PUF computations during verification. Allowing such errors, however, will make the adversary's task easier.

The probability that at most t out of n PUF responses differ from the corresponding reference responses [12] is:

$$\text{prob}_v(n, t, \mu) = 1 - \sum_{i=t+1}^{n} \binom{n}{i} \mu^i (1-\mu)^{n-i}$$

On the other hand, the probability that at most t out of n responses differ from the responses of another tag is:

$$\text{prob}_f(n, t, \tau) = 1 - \sum_{j=t+1}^{n} \binom{n}{j} \tau^j (1-\tau)^{n-j}$$

For example, using the empirical values reported in [12] for PUF:$\{0,1\}^{64} \rightarrow \{0,1\}^8$ with feed-forward arbiters, $\tau = 0.4$ and $\mu = 0.02$ (in a reference environment), and taking $n = 30$ and $t = 3$, yields a valid signature detection probability of $\text{prob}_v = 0.997107$ and a forgery non-recognition probability of $\text{prob}_f = 0.000313$. These probabilities are arguably sufficient for many RFID applications. However, if greater security guarantees are necessary, n and t can be easily adjusted to increase prob_v and to decrease prob_f even further.

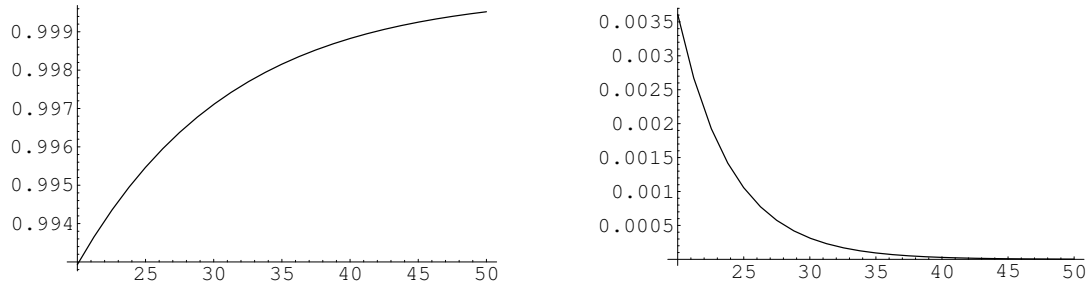
In general, the application will require $0 \leq \text{prob}_f < \beta$ and $\alpha < \text{prob}_v \leq 1$ for some reliability probability requirements α and β, and fixed n and t. Figure 1 shows the graphs of prob_v and prob_f from which appropriate n, t, α, and β can be determined. These graphs also show that increasing n and t causes prob_v and prob_f to rapidly converge to 1 and 0, respectively.[1] In these graphs we set $t = 0.1 \cdot n$, in order to simplify the illustration.

Theorem 4.1 *Given a random oracle assumption for PUF p, the probability that an adversary can forge a signature σ for a message m is bounded from above by β.*

Proof sketch: To forge a signature for a message m, an adversary must find n distinct numbers $r_1, \ldots, r_n$ as well as an unused counter value c, and compute the correct PUF values

[1] Note that the number t of PUF responses allowed to be incorrect must be a growing function of the number of challenge-response iterations n. That is, the more challenges a tag has to respond to, the more errors it is allowed to make, otherwise prob_v will approach 0.

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE



(a) $\mathrm{prob}_v(n)$ (b) $\mathrm{prob}_f(n)$

Figure 1. Graphs showing (a) the valid signature detection probability prob_v, and (b) the forgery non-recognition probability prob_f, as functions of the number of challenges n. Following [12], we set the tag uniqueness probability $\tau = 0.4$, and the reliability probability $\mu = 0.02$. We also fix $t = 0.1 \cdot n$. The two functions are plotted on different vertical scales to better illustrate their behavior. Note that the valid signature detection probability prob_v quickly converges to 1, and the forgery non-recognition probability prob_f quickly converges to 0, as a result of only a modest increase in the number of challenges n.

$p_c(r_i, m)$ for at least $n - t$ of them. Since p is assumed to be random and c was never inputted into p, an adversary can do no better than to rely on the tag(s) in its possession. □

Since the message space is large, the key (i.e., the number of message-response pairs needed to uniquely specify a PUF's behavior in digital form, assuming a PUF is difficult to model) is too long to compute and store in the back-end database. Therefore, a tag must be within range of the verifier for MAC verification. The signature is verified by asking the tag to recompute the value of p_c on input (r, m) for each r generated by the tag during the signing algorithm. If at most t out of n responses are incorrect, the signature is assumed to be authentic, otherwise it is a forgery.

The signature verification can be authorized by a tag based on a special password sent to the tag by the verifier. The verification can proceed in the clear (i.e., be observable by an adversary) if it is a one-time operation. Otherwise, the verification should be performed securely (e.g., take place inside a radio-protected environment such as a "Faraday cage"). The password-based verification mechanism provides a back-door (side-channel) for an adversary to forge a signature, and thus must be protected. We assume that two passwords can be compared without leaking any side-channel information. For example, it is important to compare passwords in their entirety rather than bit-by-bit, to prevent relatively simple power analysis attacks [13].

Observe that in this algorithm, a PUF is analogous to a public key. The algorithm for a PUF computation (i.e., the PUF schematic) is known, but the private key (i.e., a PUF's input/output behavior) remains unknown. In our buyer/seller example, even though the seller possesses a tag, he cannot predict the tag's PUF computations. Private key cryptography cannot offer an equivalent solution without relying on trusted parties. This PUF property can also be leveraged to achieve private tag ownership transfer.

The MAC protocol discussions above disallowed hardware tampering. This restriction was imposed because a physical attack on the MAC protocol may allow an attacker to steal the digital password to the MAC verification algorithm, allowing him to forge a signature. We can defend against such physical tag tampering by physically locating the tag's verification password storage circuitry below the PUF's circuit/wires. This strategy will cause an invasive attempt to recover this password to physically alter (or even destroy) the PUF with high likelihood.

Similarly, the source of the message should also be protected by the PUF. However, even with such physical protection, the verifier will not know if tag tampering occurred during verification, since the verifier does not have knowledge of any secret information aboard the tag besides the verification algorithm password. Therefore, in order to detect forgeries, the verifier must learn some information about the PUF before the tags are deployed. Specifically, it may learn enough information to authenticate the tag, as discussed above. Thus, the signature is verified only after successful tag authentication.

4.4 Small Message Spaces

If the set of messages that may need to be signed is relatively small and known a priori, PUF outputs can be computed for a selected set of tokens for all of the messages. The tokens generated aboard a tag prevent adversarial replays of signatures, and the tokens cannot be random, in order to allow signature verification without a tag's presence. Consequently, as in the MAC protocol discussed above, counters aboard a tag can serve as tokens. Since counters are part of the signature, some private information about the tags may leak out to an adversary. Next, we discuss the Key Generation, Tagging, and Verification components of the MAC protocol.

During Key Generation, the verifier creates a table of PUF values for each tag and for all possible message and counter values. The Tagging algorithm signs a message and the Veri-

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

fication algorithm verifies the signature. The key generated by the back-end system is necessarily large to enable verification without the tag's presence. As before, construction of the key occurs in a secure environment before the tags are deployed. The ability to construct keys can subsequently be disabled by short-circuiting certain wires. Alternatively, a special password can be used to control access to the key learning process, and a key can be recomputed when necessary.

Let M be a small set of messages, and let k be the number of signatures a tag needs to produce. Let P be the set of tag/PUF identifiers, and let c denote a tag's counter. For each PUF $p \in P$, each message $m \in M$, and $1 \leq i \leq n$, we compute $p_c^{(i)}(m)$ for $1 \leq c \leq k \cdot q$, and $p^{(i)}$ denotes the composition of PUF with itself i times. This dataset K of PUF values serves as the key stored with the verifier. The signature for message m is $\sigma = (\{c, p_c(m), \ldots, p_c^{(n)}(m)\}, \{c+1, p_{c+1}(m), \ldots, p_{c+1}^{(n)}(m)\}, \ldots, \{c+q-1, p_{c+q-1}(m), \ldots, p_{c+q-1}^{(n)}(m)\})$. After signing the message, a tag will increment its counter, $c = c + q$.

Input: Message set M; tag/PUF identifiers set P;
# of needed signatures k; # of sub-signatures q
for *each PUF* $p \in P$ **do**
  for $i = 1$ **to** $|M|$ **do**
    for $c = 1$ **to** $k \cdot q$ **do**
      Key$[p, m_i, c] = \{c, p_c(m_i), \ldots, p_c^{(n)}(m_i)\}$
    end
  end
end

Figure 2. Key Generation Algorithm

Input: Message m; # of sub-signatures q
Signature $\sigma = (\{c, p_c(m), \ldots, p_c^{(n)}(m)\}, \{c+1, p_{c+1}(m), \ldots, p_{c+1}^{(n)}(m)\}, \ldots, \{c+q-1, p_{c+q-1}(m), \ldots, p_{c+q-1}^{(n)}(m)\})$
Side effect: $c = c + q$

Figure 3. Tagging Algorithm

In order to minimize an adversary's chances of mounting a successful impersonation attack, we create a composite signature consisting of a sequence of q "sub-signatures", each containing n PUF computations. To avoid storing n counters aboard a tag, we instead compose the PUF with itself n times. Since the PUF's reliability is not perfect, and each repeated PUF composition depends heavily on the output of the preceding one, an invalid result computed early in the chain (see Section 3) may invalidate all subsequent values. To address this issue, a PUF can be run several times for each input; moreover, multiple independent PUF chains (i.e., sub-signatures) can be employed. The verifier checks that at least a *threshold* number of sub-signatures are valid. The algorithms for Key Generation, Tagging, and Verification are shown in Figures 2, 3, and 4 respectively.

Input: Key K; PUF p;
# of needed signatures k; # of sub-signatures q;
allowed number t of incorrect PUF responses;
Signature $\sigma = (\{c, p_c(m), \ldots, p_c^{(n)}(m)\}, \{c+1, p_{c+1}(m), \ldots, p_{c+1}^{(n)}(m)\}, \ldots, \{c+q-1, p_{c+q-1}(m), \ldots, p_{c+q-1}^{(n)}(m)\})$
verify that $1 \leq c \leq k \cdot q$
$v = 0$
for each sub-signature σ_c **do**
  $\sigma^* = \mathrm{K}[p, m, c]$
  if σ_c agrees with σ^* in at least $n - t$ terms **then**
    $v = v + 1$
  end
end
if $v \geq$ threshold **then**
  accept
else
  reject
end

Figure 4. Verification Algorithm

Theorem 4.2 *Given a random oracle assumption for a PUF p, the probability that an adversary could forge a signature σ for a message m is bounded from above by $q \cdot \beta$.*

Proof sketch: We assume that an adversary can determine (or probabilistically guess) the counter value c that a tag will use to sign its next message. However, since the PUF is assumed to be a random function, and accurate PUF modeling is not possible, an adversary can do no better than use other tags for impersonation. The success probability of forging a single sub-signature is therefore bounded by β; similarly, the success probability of forging the whole signature is bounded by $q \cdot \beta$. □

4.5 Attacks on the MAC Protocols

Relying on the properties of PUFs (described above), we consider four possible types of attacks against the MAC protocols described, and suggest corresponding PUF-based defenses for each one.

1. *Impersonation attacks:* An adversary can try to manufacture a duplicate of a target tag and then use it to forge signatures. Alternatively, an adversary can obtain (or steal) multiple PUF-based tags, and use their responses to impersonate the PUF of the target tag. Relying on the physical properties of PUF construction, we assume that duplicating a PUF or selecting an equivalent PUF out of a large pool, is improbable (indeed, this is exactly why such functions are called "unclonable"). Moreover, increasing n and t can make the valid signature detection probability prob_v arbitrarily close to 1, and the forgery non-recognition probability prob_f arbitrarily close to 0, thus making impersonation improbable.

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

2. *Modeling attacks:* An adversary can attempt to model a PUF by observing / learning the PUF signatures for selected messages. However, the highly unpredictable factors that determine a PUF's behavior are very difficult to model [7] [8] [12]. Alternatively, an adversary can try to physically dissect a tag and use electrical testing probes to measure its internal wire delays. Such attacks will be foiled by the fact that the very delays thus attempted to be measured will themselves be significantly altered by the electrical coupling between the circuit and the measuring probes. Moreover, such an approach is likely to be very disruptive to the integrity of the RFID chip, e.g., it can easily damage the overlying circuit components while probing / measuring the underlying wires [8].

3. *Side-channel attacks:* Side-channel attacks such as timing and power analyses, among others, can attempt to determine the secret information stored aboard a tag. However, PUF-based secrets appear to be difficult to learn because they are difficult to represent both accurately and concisely in digital form, and thus are not easy to model.

4. *Hardware tampering attacks:* Hardware tampering attacks that attempt to physically probe wires run a high risk of altering (or destroying) the PUF's computational behavior. Also, attempting to physically read-off or alter the digital key/password from a tag can damage the overlying wires and alter the tag's behavior, as discussed earlier. Hardware tampering can be detected using the authentication protocol discussed above.

5 Comparing PUF with Digital Hash Functions

PUFs require much less hardware to implement than the known cryptographic hash functions (which require thousands of gates). For example, existing designs of MD4, MD5, and SHA-256 require from approximately 7,350 gates to 10,868 gates to implement [6]. It is possible to use block ciphers to implement hash functions; however, even RFID-specific implementation of the Advanced Encryption Standard (AES) requires 3,400 gates [5]. Alternatively, a special hash function design for low-power devices can be used [19], but this scheme still requires 1,701 gates for a 64-bit input size, and its security has not yet been widely accepted.

In contrast, PUF-based hash functions require fewer gates than the construction of [19]. Based on a suggested PUF circuit implementation [8], we estimate that the delay circuit requires about 6 to 8 gates for each input bit, and the oscillating counter circuit that measures delay requires about 33 gates. Therefore, a 64-bit input PUF requires only about 545 gates, considerably fewer than alternative schemes, and an order-of-magnitude improvement over standard hash functions.

On the other hand, the low hardware complexity of PUF-based hash functions has a cost. The output of a PUF is only probabilistically consistent with the expected output. Also, different copies of a PUF circuit tend to have similar computational behavior (i.e., many input-output pairs are identical). In addition, PUF-based back-end systems must have enough memory to store all the challenge/response pairs for each chip. These constraints give rise to interesting constructions.

From an adversary's viewpoint, PUFs form an attack target that is different from classical digital cryptographic systems. To break traditional keyed hash functions, adversaries attempt to determine the key of the hash function. In contrast, learning a PUF's "key" appears to be more difficult since the key is difficult to represent accurately in a concise form. Analogously to the key discovery attack on standard hash functions, one could attempt to build a model for the PUF. However, model building for PUFs seems quite difficult because the PUF circuits contain numerous built-in non-linear delay components.

Compared to their digital counterparts, PUF-based hash functions appear to be more resistant to side-channel attacks and even to physical tampering. This is due to the apparent difficulty of creating a duplicate PUF, even when all the desired physical measurements can be made. No known digital hash function has this property. Thus, PUFs are highly desirable in otherwise vulnerable RFID systems which are too cost-constrained to implement more complicated defenses. On the negative side, PUFs rely on physical characteristics which are not easy to quantify, making PUF security difficult to guarantee or characterize analytically.

6 Building Physically Unclonable Functions

The PUF design of [8] is the first known prototype of a silicon PUF. One of the weaknesses of this design is that it employs an oscillating counter circuit to measure intrinsic delays, thus requiring a long time to sufficiently separate delay values for different challenges. Such a slower counting mechanism may not be problematic aboard an RFID tag (which is idle most of the time). However, this may slow down the manufacturing process when many challenge-response pairs need to be collected for each tag, and such manufacturing delays may translate into increases in overall system cost.

The distribution of the delay values for different challenges tends to be Gaussian, with many challenges producing identical (or similar) outputs even when signals take different paths through the delay circuit. Consequently, some challenges should be avoided, which requires them to be identified and filtered out during manufacturing when the database of the challenge/response pairs is created. Also, the reliability of the PUF responses is relatively low, requiring more computation rounds, while still risking producing noise. These are serious issues that may not be critical in high-end PUF systems, but are likely to be critical for low-cost RFID tags.

In order to avoid these drawbacks, a better PUF circuit could leverage sub-threshold voltage techniques [18] to compare gate polarizations, thus running quickly without using an oscillating counter. Such methods can be expected to better separate PUF values for different challenges, and thus avoid highly skewed distributions of PUF responses while still preserving PUF reliability and unpredictability. To keep the PUF modeling difficult, variable non-linear delays can be added to the circuit [8].

Proceedings of the Fifth Annual IEEE International
Conference on Pervasive Computing and Communications (PerCom'07)
0-7695-2787-6/07 $20.00 © 2007 IEEE

7 Future Research

More work is needed in the area of hardware-based RFID to help bring the ideas presented in this paper into commercial reality. Theoretical breakthroughs would include theorems that characterize the security of PUFs. RFID tags equipped with PUFs satisfying current RFID standards need to be fabricated and thoroughly tested under different environmental and operational conditions. For example, sub-threshold voltage -based PUFs that exploit non-linear circuit behavior seem promising. The behavior of PUF-based tags should be tested under varying levels of motion, acceleration, vibration, temperature, noise, etc. In each application / scenario, the probability of output collision τ and the probability of an incorrect response μ should be characterized as functions of the operational environment.

New PUF-based RFID security protocols should be developed for different applications, including multi-tag regimes [3]. PUFs can also benefit RFID ownership transfer algorithms, and in helping to detect and negate privacy compromises in tree-based identification protocols. Utilizing PUFs aboard RFID readers (as opposed to only aboard tags) can help thwart malicious readers in the field, and make it much more difficult for adversaries to clone readers. Finally, we note that while the inherent unpredictability of PUF outputs can be exploited to permit certain protocols (as done above), their *predictability* in some instances can be exploited to create other new protocols.

8 Conclusion

Since RFID tags are power-limited and cost-constrained, it is not feasible to implement full-fledged cryptographic security mechanisms aboard RFID tags. We therefore proposed a PUF-based approach that utilizes hardware support to provide security and privacy. PUFs supply an exponential number of keys aboard the tags, thereby providing a viable practical solution to a critical key distribution problem. PUFs can also protect tags from cloning, even if an adversary has physical access to them and their circuit schematics. This property makes PUF-equipped tags valuable in access control and authenticity verification applications.

We have shown how PUFs aboard RFID tags can be used to implement tag identification while relying on hardware support and making the unauthorized tracking of tags considerably more difficult. We also described novel protocols for message authentication codes (MACs) that require little hardware resources and thus mitigate tag cost escalation. We compared PUFs to their digital counterparts and offered possible improvements in PUF design. We outlined numerous directions for PUF-related future research that include extensive testing of PUFs, proving security theorems about PUFs, designing new algorithms, and putting PUFs into RFID readers.

9 Acknowledgements

We thank Blaise Gassend and Daihyun Lim for fruitful discussions on PUF complexity, and Ben Calhoun for suggestions on how to build a better PUF. Special thanks go to the reviewers who helped to greatly improve the paper.

References

[1] G. Avoine and P. Oechslin, *A scalable and provably secure hash based RFID protocol*, International Workshop on Pervasive Computing and Communication Security, pp. 110-114, 2005.

[2] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, Proc. ACM Conf. on Computer and Communications Security, pp. 62-73, 1993.

[3] L. Bolotnyy and G. Robins, *Multi-Tag Radio Frequency Identification Systems*, Proc. IEEE Workshop on Automatic Identification Advanced Technologies, pp. 83-88, 2005.

[4] L. Bolotnyy and G. Robins, *Generalized 'Yoking-Proofs' for a Group of RFID Tags*, Proc. International Conference on Mobile and Ubiquitous Systems (Mobiquitous), 2006.

[5] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, *Strong Authentication for RFID Systems Using the AES Algorithm*, Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer LNCS 3156, pp. 357-370, 2004.

[6] M. Feldhofer and C. Rechberger, *A Case Against Currently Used Hash Functions in RFID Protocols*, Workshop on RFID Security (RFIDSEC), 2006.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, *Controlled Physical Random Functions*, Proc. Computer Security Applications Conference, 2002.

[8] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, *Silicon Physical Random Functions*, Proc. Computer and Communication Security Conference, 2002.

[9] A. Juels, *Minimalist Cryptography for Low-Cost RFID Tags*, Proc. International Conference on Security of Communication Networks (SCN), pp. 149-164, 2004.

[10] A. Juels, *'Yoking-Proofs' for RFID Tags*, Proc. International Workshop on Pervasive Computing and Communication Security, pp. 138-143, 2004.

[11] L. Lamport, *Constructing Digital Signatures from a One Way Function*, Technical Report CSL-98, SRI International, 1979.

[12] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, *Extracting Secret Keys From Integrated Circuits*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13 (10), pp. 1200-1205, 2005.

[13] Y. Oren and A. Shamir, *Power Analysis of RFID Tags*, `http://www.wisdom.weizmann.ac.il/~yossio/rfid/`.

[14] M. Philipose, J. Smith, B. Jiang, A. Mamishev, S. Roy, and K. Sundara-Rajan, *Battery-Free Wireless Identification and Sensing*, Pervasive Computing, Jan-Mar 2005.

[15] D. Ranasinghe, D. Engels, P. Cole, *Security and Privacy: Modest Proposals for Low-Cost RFID Systems*, Proc. Auto-ID Labs Research Workshop, Zurich, Switzerland, September 2004.

[16] P. Ravinkanth, *Physical One-Way Functions*, Ph.D. Thesis, MIT, 2001.

[17] P. Tuyls and L Batina, *RFID-Tags for Anti-Counterfeiting*, Topics in Cryptology (CT-RSA), The Cryptographers' Track, RSA Conference, pp. 115-131, 2006.

[18] A. Wang, B. Calhoun, A. Chandrakasan, *Sub-threshold Design for Ultra-Low Power Systems*, Springer, 2006.

[19] K. Yuksel, *Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications*, Master Thesis, Worcester Polytechnic Institute, Massachusetts, 2004.