Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions

Nathan Beckmann[1] and Miodrag Potkonjak[2]

[1] Massachusetts Institute of Technology
beckmann@csail.mit.edu
[2] University of California, Los Angeles
miodrag@cs.ucla.edu

Abstract. A physically unclonable function (PUF) is a multiple-input, multiple-output, large entropy physical system that is unreproducible due to its structural complexity. A public physically unclonable function (PPUF) is a PUF that is created so that its simulation is feasible but requires very large time even when ample computational resources are available. Using PPUFs, we have developed conceptually new secret key exchange and public key protocols that are resilient against physical and side channel attacks and do not employ unproven mathematical conjectures. Judicious use of PPUF hardware sharing, parallelism, and provably correct partial simulation enables 10^{16} advantage of communicating parties over an attacker, requiring over 500 of years of computation even if the attacker uses all global computation resources.

Keywords: PPUF, security, cryptography, public key cryptography.

1 Introduction

Motivation. Cryptography is a scientific and engineering field that develops and analyzes techniques for protecting privacy of stored or communicated information. Currently, it is mainly realized using secret key (a.k.a. symmetric key, shared key, private key, and one key) and public key techniques. The emergence and rapid proliferation of mobile, sensing, health, financial, e-commerce and other pervasive applications has elevated the system importance of sound, practical cryptographic protocols.

Cryptographic techniques, and in particular public key protocols, have been the basis for numerous security applications, ranging from secure email, secure remote access (e.g. passwords and smart cards), remote gambling, and digital signatures to privacy protection, digital rights management, watermarking, and fingerprinting. However, there are two major drawbacks of classical cryptographic techniques that are widely documented in security literature. The first is that the current state-of-the-art cryptographic techniques are based on extremely likely but nevertheless unproven mathematical assumptions. The second is that even if there are no algorithmic weaknesses in public key cryptographical protocols, often they can be easily broken due to software vulnerabilities, physical attacks, or side channels.

Our primary goal in this paper is to present a new type of cryptography that uses a generalized form of physically unclonable functions (PUFs), an approach that resolves

S. Katzenbeisser and A.-R. Sadeghi (Eds.): IH 2009, LNCS 5806, pp. 206–220, 2009.

the two main conceptual and practical limitations of classical public cryptography. The first disadvantage, the use of non-proven mathematical conjectures, is replaced with technological, physical, and chemical laws that prevent manufacturing of fully identical physical systems at gate and transistor levels of silicon technology or other nano-scale systems. The second, more important vulnerability, susceptibility to physical and side channel attacks, is completely eliminated. In addition, PPUF-based security is in many applications much faster and requires significantly less energy. For example, in remote authentication, the new scheme requires only one control cycle to generate the correct answer.

The new cryptographical approach is based on the novel notion of a public physically unclonable function (PPUF). A PUF is a physically system that is intractably complex to replicate. Modern and future silicon technology-based integrated circuits may serve as PUFs due to their intrinsic manufacturing variability. PUFs have been manufactured and their use for secret key-based security applications has been demonstrated. Recently, it has been shown that many types of PUFs can be easily reversed engineered. While these approaches jeopardize some secret key applications of PUFs, they also create starting points for the creation of public key-based protocols that employ unclonable hardware.

PPUFs form a class of PUFs that can be reversed engineered, but once their structure is completely characterized, one still requires very large time to compute the PPUF outputs for a given input. We use PPUF characteristics as a public key. We focus on PPUFs realized as a small circuit. It is relatively easy to envision how PPUF with the ratio of simulation vs. execution times of κ, gives the computational advantage of κ to each of the communicating parties (see below). What is remarkable is that this advantage can not only be used for remote exchange of secret information, but can be further significantly amplified using parallel computations and directly used for creation of a number of security protocols. We believe that PPUF-based cryptography will provide an impetus for the creation of conceptually new security approaches that are not just much faster and use much less energy, but are also resilient against physical attacks and side channels.

The paper is organized as follows. In §2, we discuss the related work in crytography and circuits necessary for description of PPUFs. In §3, we discuss some preliminary results that our work is based on. In §4, we present and analyze a PPUF architecture. In §5, we present a secret key exchange protocol using PPUFs. And §6 concludes the paper.

A Simple Example. We now demonstrate the operation of a PPUF and how it can be used in a secret key exchange protocol. Figure 1 shows a simple PPUF consisting of 6 XOR gates arranged in three rows. The delay through each gate is also shown in Table 1. Due to manufacturing variability, the delays are unequal for each gate and each input (see §3).

We assume that "01" is initially on the input and the circuit has reached a steady state (output "00" on gates E, F). Then, at time $t = 0$, the input becomes "10". At $t = 0.88$, the "0" reaches the output of gate B, which becomes 0. At $t = 1.12$, the "1" reaches the output of gate B, and its output becomes 1. Similarly, at $t = 0.93$ and $t = 1.01$, gate A transitions to 0 and then 1.

Table 1. Gate delays from given input to output (in ps)

	Input 1	Input 2		Input 1	Input 2
E	.86	.95	F	1.24	.96
C	1.11	.90	D	.78	.71
A	.93	1.01	B	1.12	.88

Table 2. Output transitions

E	$\{2.54, 2.64, 2.66, 2.74, 2.78, 2.88, 2.9, 2.98\}$	F	$\{2.55, 2.67, 2.75, 2.79, 3.02, 3.26, 3.28, 3.36\}$
C	$\{1.78, 2.02, 2.04, 2.12\}$	D	$\{1.59, 1.71, 1.79, 1.83\}$
A	$\{.93, 1.01\}$	B	$\{.88, 1.12\}$

This pattern repeats through each row of the PPUF. On row 2, gate C transitions each time a new input arrives. Looking at Table 1, this occurs at transitions of gate A plus 1.11 ps and transitions of gate B plus $.90$ ps. All in all, gate C transitions at times $t \in \{2.04, 2.12, 2.02, 1.78\}$. Similarly, gate D's output transitions whenever gates A or B transition, plus the delay through gate D. Gate D transitions at times $t \in \{1.71, 1.79, 1.83, 1.59\}$. It should be clear that on row 3, gates E and F will each transition 8 times, when either C or D transition (plus the delay through the gate). This gives rise to an exponential number of transitions on the number of rows (§4.2). See Table 2.

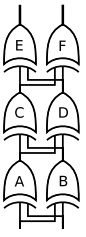

Fig. 1. A simple PPUF

We now show how to exchange a secret key between two parties, Alice and Bob. Suppose Alice owns the above PPUF. The gate-level characterization of the circuit (given by Table 1) is effectively the public key, enabling accurate simulation of the PPUF. Bob begins by choosing two numbers to input into the circuit — suppose he chooses $x_0 = 01$ and $x_1 = 10$. Bob also chooses a time, say $t = 2.7$ ps. Bob then simulates the PPUF starting at steady state on input x_0 with input x_1 arriving at time 0, attempting to determine the output after 2.7 ps. To do so, he computes all 16 output transitions as we have done, concluding that the output reads $y = 10$ at 2.7 ps.

Bob then sends x_0, t, and y to Alice. It is now Alice's job to find x_1. To do so, she iterates over all possible inputs and checks the output of the PPUF for each, clocking the output at $t = 2.7$ ps. In this case, $x_1 = 10$ is the only input that produces output y after 2.7 ps (see Table 3). In a sense, the PPUF is the private key, enabling Alice to quickly find x_1 — the PPUF runs in a matter of picoseconds, so searching the entire input space takes little time.

An attacker, on the other hand, gets the worst of both worlds: he must simulate every possible input until x_1 is found. He is at a disadvantage over Alice of simulating the PPUF instead of running it (see §4). He is at a disadvantage to Bob of simulating several values instead of a single one (see §5). Expanding on these two advantages, we are able to achieve insurmountable advantage over an attacker.

2 Related Work

Cryptography. Since the mid seventies when the first paper on public key cryptography [9] and the first practical realization [30] were published, cryptography has developed into a large field with a wide variety of elegant results. A number of excellent books are available [24] that emphasize both the theoretical [12] and practical [32] aspects. Numerous public key cryptography protocols have been developed for ciphers, hash functions, message digests, message authentication codes, asymmetric public key secure communication and storage, public key infrastructure, digital signatures, authentication, zero knowledge proofs, secret sharing, digital money, secure watermarking, remote gambling, and many other applications. More recently, quantum cryptography has been attracting a great deal of interest.

Table 3. Output at 2.7ps

Input	Output
00	01
01	00
10	10
11	11

Side Channel Attacks. However, quantum cryptography is still a pending technology, and there are several serious problems with traditional mathematical cryptography. While there is relatively little chance that its unproven foundations will be compromised, and although conceptually new algorithmic and statistical attacks are rare and often fixable [5], there is a wide consensus that a great variety of often inexpensive and fast physical and side channel attacks are surprisingly effective [17,34].

Unclonable artifacts and PUFs. Unique and unclonable artifacts were first proposed in early eighties [3]. More recently, several actual implementations have been demonstrated and analyzed [6]. Lofstrom et al. proposed use of silicon manufacturing variability (MV) as a source of unique integrated circuits [20]. Koushanfar et al. proposed used of unique integrated circuits as security and digital right management mechanisms [19]. Papu et al. introduced powerful notion of physical one-way functions [28]. Devadas and his research group realized that silicon MV is a very practical technology for creation and use of such physically unclonable functions (PUFs) and demonstrated and analyzed their properties [11]. The joint efforts of Rice and UCLA system security groups demonstrated a number of techniques for exposing vulnerabilities of a wide classes of initial silicon PUFs [22], and introduced several secure PUF architectures [21].

Timing Precision. There are a large number of techniques for the creation and measurement of rapidly changing signals: (i) interval stretching followed by digital counting, (ii) time-to-amplitude conversion combined with A/D conversion, and three purely digital methods, (iii) the Vernier method with two oscillators, (iv) time-to-digital conversion using a tapped delay line, and (v) the Vernier method using two differential delay lines [16]. Although progress has been steady, it is difficult to achieve time resolution significantly better than 1 picosecond using standard on-chip technology and

techniques. However, with the use of lasers and materials that change their properties under the impact of changing light, methods exist for measurements in femtosecond and attosecond range [13,2]. Currently, the most accurate clocks measure time intervals in a range of 10 attoseconds.

3 Preliminaries

In this section, we briefly summarize the sources of manufacturing variability (MV), MV modeling, and its impact on delay, dynamic, and leakage power of a gate, issues and techniques used for gate-level characterization, and introduce our approaches for addressing time variability of these characteristics due to operational and environmental impact.

Manufacturing Variability. As the feature size of silicon integrated circuits keeps shrinking, any given gate from a single design has unique characteristics on each physical implementation of the design [4]. Essentially, a number of unavoidable physical and chemical phenomena, such as silicon lattice imperfections, uneven distribution of dopants, imperfect mask alignment, and non-uniform chemical mechanical polishing, result in gates with sharply different characteristics [31]. Already in 45 nanometer technologies, it is common that the delay of the same gate in different ICs differs by 1/3 from the nominal value and that leakage power differs by factor of 20. Note that while in 1 micron technology, each transistor had a million dopants, in 45 nanometers, the number is only a few hundred [31]. Therefore, even small variations have pronounced impact. In future technologies, this situation is bound to become even more significant. In addition, if the goal is to intentionally create high and unreproducable manufacturing variability, variable exposition to strong light can further enhance MV by at least two orders of magnitude [26].

Gate-level Characterization. Gate-level characterization is a process of characterizing each gate of an IC in terms of its physical properties such as gate width, gate length, and thickness of oxide or its manifestation properties such as delay, leakage power, and switching power. An important observation is that if the physical properties are known, it is straightforward to calculate the manifestation properties and vice versa [23]. There are two main and orthogonal approaches for gate characterization. The first focuses on direct measurement of physical parameters using sophisticated microscopes [10]. The second one measures global delays between flip-flops or static or dynamic power for different input vectors and uses various techniques for solving systems of equations under various assumptions to find individual gate characteristics [33,18,1].

The advantage of the first approach is that one can directly measure all gates on each IC regardless of design structure. However, the approach is very expensive and slow and requires wafer-level inspection, which has significant potential for damage. On the other hand, the second approach is fast, inexpensive, and can be applied on packaged ICs, but sometimes a fraction of gates can not be characterized.

While these efforts address gate-level characterization of an arbitrary IC, Dabiri et al. [8] have developed a specialized architecture that enables complete and accurate characterization of all gates even in presence of significant error measurements. Their technique combines statistical modeling and convex programming for very accurate gate-level characterization. Our architecture design is even better suited to this technique than their requirements, so it is assumed this technique is used for characterization.

Stability. Increases in temperature may significantly increase the delay of pertinent gates. Supply voltage has even greater impact, quadratic on switching power and linear on delay [23]. The surrounding environment and operational conditions may significantly alter the nominal manifestation parameters of each gate, sometimes even in different ways for different gates.

In order to preserve the correctness of our hardware-based cryptography approach, we use three approaches, two synthetic and one operational. First, we place all gates as close as possible and supply them by the same part of the power/ground networks so that the differential impact of manufacturing variability is minimized. Second, we place several delay paths that consist only of inverters and multiplexers that can be rapidly characterized interleaved with the PPUF circuitry. Our cryptographic procedure is invoked only when no or consistent delay changes are detected on these delay paths.

4 Public Physically Unclonable Functions (PPUFs)

We begin with a description of a PPUF and its operation. Later sections demonstrate the cost of simulation in the general case, how to ensure correct operation under a variety of conditions, and technological trends related to PPUFs.

4.1 Description

As defined earlier, a PPUF is a physical system that is unclonable due to its structural complexity, yet whose simulation is feasible, although requiring large time to do so. We describe how to create a simple circuit meeting these criteria.

A PPUF is a rectangular array of gates of size w x h (Figure 2). Inputs are fed into the bottom row, and outputs are read from the top row. Each intermediate row feeds the next, with each gate having b inputs from the previous row.

Due to manufacturing variability, the delay through each gate will vary by a significant percentage from its neighbors. Furthermore, because of the transistor-level construction of gates, the delay through any given gate for each of its inputs will differ. Thus, as demonstrated in §1, there will be many transitions on the output before the circuit reaches steady state.

In order to operate a PPUF, we need three values: x_0, the previous input; x_1, the input; and t, the output time. We assume that the circuit has reached steady state with input x_0 before x_1 arrives. x_1 is input to the circuit, and we clock the circuit at precisely time t to read the output. This is the final output of the PPUF.

This circuit is a physically unclonable function (PUF) because its output is heavily dependent on manufacturing variability in the delay of its gates. This is inherently unfeasible to replicate with the same manufacturing technology. This circuit is public, however, because given the delay of each gates it can be simulated. As we'll see in the next section, this requires exponential time in the height of the circuit.

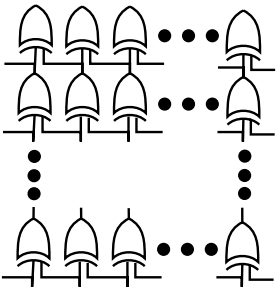

Fig. 2. A canonical PPUF circuit

4.2 Simulation Analysis

We now analyze the simulation cost of a PPUF. We first do a simple analysis assuming all gates are XORs. Then we present a serious flaw with using XOR/XNOR gates and how to negotiate that issue.

Cost Analysis Using XOR. Consider the simulation of the PPUF on input x_1. We must first calculate the state of the circuit when x_0 arrives. Because the circuit has reached steady state, this computation is trivial to perform by going through every row of the PPUF, starting at the input, and computing its output based on the output of the previous level. This can be done in linear time on the number of gates. But because the output of the PPUF is measured before the circuit reaches steady state, simulation is no longer so simple.

We proceed using dynamic programming to compute the timings of transitions for every gate at each level. Because the gates are XORs, a transition on the input will correspond to a transition on the output. We calculate the output level-by-level, as before, except now we must track much more information. We must record every transition that occurs at each level, because the timing that ultimately appears at time t could originate from any of large number of timings from intermediate gates. (There are exponentially many paths from an intermediate gate to an output gate, each with unique delay.)

We therefore measure the simulation cost of a PPUF as the number of transitions on the output and intermediate gates. We assume that x_1 and x_0 are independently selected — that is, the probability of transition between any bit of x_1 and x_0 is $\frac{1}{2}$. Let N_i be the number of transitions at row i in the PPUF. Let n_i be the number of transitions for a *single* gate at level i.

$$N_i = wn_i \tag{1}$$

$$n_i = bn_{i-1} \tag{2}$$

Since the delay from each input of a *single* gate to the output is unique and inputs are random, $\mathbb{E}(n_0) = \frac{1}{2}$. Finally, the expected number of transitions at level i is,

$$\mathbb{E}(N_i) = \frac{1}{2}wb^i \tag{3}$$

The simulation cost of the entire PPUF is measured at level h, $\mathbb{E}(N_h) = \frac{wb^h}{2}$. The simulation cost is exponential in the height of the PPUF. This model agrees extremely well with simulation under the above assumptions.

Table 4. "Good" 3-input gates

Input	A	B	C	D
000	0	0	1	1
001	0	1	0	1
010	1	1	0	0
011	0	0	0	1
100	1	1	0	0
101	0	0	1	0
110	1	1	1	1
111	1	0	1	0

(a) Truth tables

Value	A	B	C	D
B_1	0.5	0.5	1.0	1.0
B_2	0.5	0.5	0.5	0.5
B_3	0.5	1.0	0.5	0.5
B	1.5	2.0	2.0	2.0

(b) Transition rates

Problems with XOR. Until now, we have made a dangerous assumption: gates toggle their output whenever there is a transition on their input. This assumption is valid for XOR and XNOR, but it allows for more efficient simulation of the PPUF, ruining our exponential advantage.

The idea is that if the output toggles at every input transition, then we can easily compute the output transitions for any input based on the transitions generated by each bit of the input. These can in turn be precomputed and sorted, giving logarithmic search time per input. This reduces the simulation time to linear on the size of the circuit.

Let T_i be the set of output transition timings that occur when only input i toggles. This is well-defined because although the output may vary, the timing of transitions will be the same regardless of the previous state of the circuit. Let $a = a_1 ... a_w$ be an input to the PPUF. (Assume that previously the input was 0, so $a_i = 1$ if and only if the i^{th} input toggles). Then the set of output timings generated by a is $T_a = \cup_{i:a_i=1} T_i$.

We can now use T_a to compute the output at any time after a arrives. First, we compute the output before a arrives in linear time. Then, to compute the output at time t after a arrives, we must know the number of output transitions occurring before t in T_a. This can be computed by summing the number from each T_i. Assuming that each T_i has been precomputed and sorted, this takes $O(\log(b^h)) = O(h \log(b))$. There are w inputs, giving overall simulation cost of $O(wh \log(b))$.

Fortunately, this approach is easily foiled. We can break the key assumption by using a more complicated truth table than XOR. We design the gate so that it has equal numbers of 0's and 1's on the output — this keeps the probability of each $\frac{1}{2}$ for each output, uniformly dividing the PPUF's output through the number space[1]. However, the only 2-input gates that have this property are XOR, XNOR, and functions of a single input, so we must move to at least 3-input gates.

There are many 3-input gates that meet our needs of (i) having equal numbers of 0's and 1's on the output, (ii) depending on all inputs, and (iii) having "complex" output behavior (not toggling on every input transition). Table 4a gives some examples.

[1] This isn't completely true, because gates share inputs and therefore aren't independent. Simulation indicates that before the circuit reaches steady state, it is still approximately correct.

Revised Cost Analysis. We must reassess the cost analysis if we use gates that do not always toggle on an input transition. Previously, in Eq. (2) we multiplied n_{i-1} by b because we assumed each input transition produced an output transition. We must replace this with the *expected number of output transitions per input transition* for the gate.

Define B_i as the expected number of output transitions when input i is toggled, and C_i the number of possible transitions if input i is toggled. Clearly, $B_i = \frac{C_i}{2^b}$.[2] For XOR and XNOR, $C_i = 2^b$ and $B_i = 1$ for all i. Now define B as the expected number of output transitions per input transition, $B = \sum_{i=1}^{b} B_i$. Eq. (2) and (3) become (assuming homogenous gates),

$$\mathbb{E}(n_i) = Bn_{i-1} \tag{4}$$

$$\mathbb{E}(N_i) = \frac{1}{2} w B^i \tag{5}$$

Our advantage remains exponential, although slightly diminished. Table 4b gives values of B_i and B for gates in Table 4a. We wrote a simulator for PPUFs of several gate types, and results agreed with the growth rates calculated for each type of gate.

4.3 Limiting Simulation Cost

This section describes a small modification to the PPUF that allows for much less expensive simulation for the simulating party in the secret key exchange protocol (§5).

As indicated in the example (§1), it will be useful to find ways to limit the cost of simulation, so long as doing so does not proportionally reduce the simulation time for an attacker. One such method is to compute a single output gate of the PPUF instead of the complete output. This requires computing a large fraction of the previous rows, but since simulation cost increases exponentially with the height of the circuit, we still save the majority of simulation cost.

This gives us a single bit as the output of the PPUF. This isn't sufficient for our purposes, because we need to be able use the output of the PPUF to distinguish between many different inputs. In order to increase the size of the output, we can simply include the output of several previous rows feeding the final output (Figure 3). These will be mapped through a hash function so that their inclusion does not provide third parties any additional information or enable them to short-circuit the simulation.

Optimistically, we are computing one of w outputs, so we should expect $\frac{1}{w}$ reduced simulation cost. But this ignores the cost of simulating the previous rows, which reduces the savings. Summing over the previous rows, it can be shown that the reduction is roughly $\frac{2}{w}$.

[2] Because we have equal numbers of 0's and 1's from each gate and the PPUF's input is assumed to be random, we can assume that the input to each gate will also be random. So for each possible input transition, we can average over all possible inputs.

4.4 Fault Tolerance

A major practical concern in the operation of PPUFs is the timing of the output. With exponential growth in the number of output transitions and the extremely fast operation of circuitry, the mean time between output transitions is tremendously small. Section 2 discussed the state of the art in measurement accuracy. This section presents a few algorithmic techniques to mitigate the problem.

There are a number of strategies to increase our ability to clock the output at the proper time. The circuit can be clocked multiple times (both simultaneously and in separate computations), with each measurement "voting" on the final value of the output. This reduces the impact of changing conditions, such as temperature.

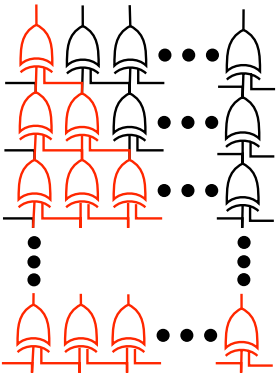

Fig. 3. Inputs "feeding" a PPUF output

More significantly, note that the simulating party is at a distinct advantage. Because the output timings are somewhat randomly distributed, there will be intervals that are much larger than the mean interval time. The simulating party can choose a time that has an unusually long stable interval.

Finally, note that the maturation of developing technologies (§2 and [13,7,2,14,25]) could eliminate this problem as a practical consideration entirely. With accuracy in the attosecond range, combined with the techniques already mentioned, the simulation cost of a PPUF could be raised as high as practically desirable.

4.5 Technological Trends

Any security techniques must be evaluated against technological trends. Although any strategy for predicting the future is inherently risky, it appears that essentially all technological trends favor hardware-based cryptography and security techniques. First of all, rapid progress in high-speed imaging technologies and micro/nano sensing will make side channel attacks even more effective and widespread. Also, the proliferation of horizontal integrated circuit business models where design tools, design, manufacturing, testing, and integration are done by different untrusted companies will increase the danger of hardware Trojan horses and hidden channels. Finally, application trends favor mobile systems where their physical security will be significantly lower.

We believe that six technology trends will have major impact on PPUFs and our new cryptography and security approach: higher levels of VLSI integration, smaller feature sizes, increasingly difficult cooling, new interconnect and input/output technologies, and more economically viable and accurate time measurement technologies.

Higher integration levels have two main ramifications: larger PPUFs and faster simulation. The main consequence is a rapidly increasing execution-simulation gap due to increased PPUF parallelism. Finer feature sizes have three qualitatively novel ramifications: the transistor will not be additionally faster, switching power will begin to increase slowly, and leakage power will become dominant [35]. The net consequence is

that thermal management will become more difficult and will effectively reduce simulation speed. New interconnect technologies such as nanowires, photonic crystals, plasmonics, RF, and 3D ICs have potential to increase I/O and data transfer rates by several orders of magnitude and, therefore, favor simulation. However, photonic crystals [29] and plasmonic wires [27] are even more sensitive to manufacturing variability, operate on much higher frequencies, are more complex to simulate, and will greatly increase the execution-simulation gap. In summary, new technologies will keep increasing the simulation speed and PPUF realized in today's technologies will be more susceptible to simulation attacks. At the same time, PPUF-based techniques will become even more attractive as technology progresses.

5 Secret Key Exchange with PPUFs

5.1 Description

We enable two parties, A and B, to exchange a secret key. We assume that A is in possession of a PPUF. B is the "simulating party" who uses the public description of A's PPUF to simulate its output for some input.

The basic idea of the protocol is simple: We consider a range of numbers of size n. B selects some number, x, from this range and simulates the PPUF on this input. B sends the output, y, to A. A then searches through all n numbers until x is found. The full protocol has a few twists to make it work correctly with PPUFs as they are described above and to get a few other useful properties.

Before describing the detailed protocol, we go over a few fundamentals. One way this protocol could be attacked is to precompute the output of the PPUF for every possible input. This is easy to prevent by choosing the secret key, x, to be a fairly long number, say 1024 bits. This would require 2^{1024} bits of storage, which is completely infeasible[3].

Another concern is that we use the output of the PPUF to distinguish between different x, but there might be collisions, leading to false matches and incorrect results. This is addressed in the same fashion by using a long enough output so that the probability of a collision is negligible[4]. Both technique come at very little hardware cost in the PPUF.

The protocol is as follows:

1. B simulates values.
   (a) B randomly selects x_0 from $0...2^w$, where w is input width of the PPUF.
   (b) B selects $x_1...x_m$ from $x_0...x_0 + n$, where n is computed as in §5.2.
   (c) B applies a hashing function, f, to compute $z_1 = f(x_1)...z_m = f(x_m)$.
   (d) B simulates $z_1...z_m$ on A's PPUF, starting with x_0 as the initial input, and timing at $t_1...t_m$. This produces outputs $y_1...y_m$.
2. B sends x_0, m, n, $y_1...y_m$, and $t_1...t_m$ to A.
3. A finds $x_1...x_m$.

[3] Actually, it requires much more. One would need to store the PPUF output for all inputs of the form (x_0, x_1, t).

[4] The output of each bit of the PPUF is 0 or 1 with equal probability, as described earlier (§4). Therefore the output is distributed uniformly through the output space, and increasing the length of the output exponentially drops the chance of collisions.

   (a) A iterates over each $x \in (x_0, x_0 + n)$.
   (b) A computes $z = f(x)$.
   (c) A runs the PPUF with x_0 as the steady-state input, z as the input, and clocking at each $t_1 ... t_m$.
   (d) If the output at time t_i equals y_i, then store x as x_i.
   (e) Halt when all $x_1 ... x_m$ are found.
4. A and B concatenate $z_1 ... z_m$ to form the secret key.

The two new elements of the protocol are the multiple values, m, and the hashing function, f. We use multiple values in order to reduce the variance in the protocol. When a single value is sent, then the search time for A or any attacker has large variance. This is undesirable if we are trying to achieve a specific level of security or allocate a set amount of work for A. By sending more values, we increase the expected fraction of the number space that must be searched and, more importantly, significantly reduce the variance.

We also apply a hashing function, f, to each value before sending it to the PPUF. This is because using partial simulation (§4.3), the PPUF's output doesn't depend on all of its inputs. By selecting from a range $x_0 ... x_0 + n$, we will have many bits that are shared between numbers. Therefore, the output of the PPUF might no longer be unique, greatly increasing the odds of collisions on the output. By applying a hashing function, we ensure that the bits of the input will be different for each x_i and therefore the output of the PPUF will be unique (with extraordinarily high probability). There are many ways of achieving the same effect — for example, defining $x_i = f^i(x_0)$, or having the output of x_{i-1} be the steady-state input for x_i.

5.2 Analysis

The advantage of this protocol comes because an attacker does not know which values have been selected, nor does he have A's PPUF to enable fast searching. He therefore must search the $x_0 ... x_0 + n$ values, simulating each, to find each x_i. Even with fairly small m, he will have to search the majority of the n numbers. So his disadvantage over B is roughly n, and his disadvantage over A is roughly κ, the cost of simulation.

More specifically, let W_A be the expected work for the owner of the PPUF. Here, work is normalized to the cost of computing the output of the PPUF. Similarly, W_B is the expected work for the simulating party, and W_O is the expected work for an observer (attacker). If $W_A \neq W_B$, then the effective computational advantage over an attacker is the minimum of either advantage. This imposes the constraint $W_A = W_B$.

The owner of the PPUF's work is dominated by the search for $x_1 ... x_m$. So W_A is simply the amount of numbers that must be searched to find all x_i. Using simple probability, $W_A = \frac{m}{m+1} n$. Similarly, the simulating party's work is dominated by simulation and $W_B = m\kappa$. This yields,

$$n = (m + 1)\kappa \tag{6}$$

The work for an attacker is the same expected number of computations, except each is a simulation of the PPUF.

$$W_O = \frac{mn}{m+1} \cdot \kappa = \frac{{W_A}^2}{m} \tag{7}$$

The attacker performs quadratically more computation than either communicating party.

Including partial simulation (§4.3), W_B is reduced by a factor of $\sim \frac{2}{w}$. This changes the results to,[5]

$$n = \frac{2}{w}\kappa(m+1) \tag{8}$$

$$W_O = \frac{mn}{m+1}\kappa = \frac{w}{2}\frac{{W_A}^2}{m} \tag{9}$$

Finally, note that the network requirements of this protocol are minimal. We send $2m+1$ values of length w and two fairly small integers. Because m is fairly small, network requirements are no more than 1 kB.

5.3 Practical Performance

This section gives numbers to the equations above and shows the advantage that can be practically gained. Pessimistically assuming that the PPUF operates in the cycle time of a general purpose processor, simulation takes κ cycles. Due to inherent parallelism in the simulation and the availability of multicores, the simulating party should have roughly 10 GHz, or 10^{10} cycles per second of computational power.

If $m = 3$ numbers are simulated and the simulating party takes 10^3 seconds (fifteen minutes) to simulate, then this gives the simulating party roughly $3 \cdot 10^{12}$ cycles of simulation per number. Assuming a PPUF with $w \approx 10^4$ (much less than could be achieved with modern silicon manufacturing technology), the simulation cost is $\kappa \approx 1.7 \cdot 10^{16}$ cycles[6]. The owner of the PPUF should search $n \approx 10^{13}$ numbers, which is obvious since $W_A = W_B$, and the simulating party spends $10^{10} \cdot 10^3 = 10^{13}$ cycles. The attacker must perform $W_O = 1.7 \cdot 10^{29}$ cycles of simulation on average to find the secret key.

Studies indicate that there are no more than one billion computers in the world [15]. The vast majority of these computers are incapable of 10 GHz of computation, but for argument we will assume that each is as powerful as the simulating party's. Then the computational throughput of an attacker has an upper bound of 10^{19} cycles per second. An attacker would take $1.7 \cdot 10^{10}$ seconds, or 528 years to break this protocol.

Throughout this example we have been extremely generous to the attacker. In reality, the security of the scheme is several orders of magnitude better than claimed.

6 Conclusion

We have developed a new approach for exchange of secret keys and public key cryptography. We use the novel notion of a public physical unclonable function and its

[5] Since $m \ll n$, the majority of simulations performed by an attacker are unsuccessful and he must simulate all outputs, increasing the cost by a factor of two. This also accounts for the reduction in W_B.

[6] Note that this does not exactly correspond to $N_h = \kappa$ (see Eq (3)), as there are constant factors that require much more than one cycle to process a single transition in the PPUF.

implementation via integrated circuits implementation where easily measured delays of PPUF gates serve as public key. The approach is intrinsicly resilient against physical and side channel attacks due to physical laws and technological constraints that prevent PPUF cloning.

References

1. Alkabani, Y., Massey, T., Koushanfar, F., Potkonjak, M.: Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability. In: Design Automation Conference, pp. 606–609 (2008)
2. Baltuska, A., Udem, T., Uiberacker, M., Hentschel, M., Goulielmakis, E., Gohle, C., Holzwarth, R., Yakovlev, V., Scrinzi, A., Hansch, T., Krausz, F.: Attosecond control of electronic processes by intense light fields. Nature 421, 611–615 (2003)
3. Bauder, D.: An anti-counterfeiting concept for currency systems. Technical report, Sandia National Labs, Albuquerque, NM (1983)
4. Bernstein, K., Frank, D., Gattiker, A., Haensch, W., Ji, B., Nassif, S.R., Nowak, E., Pearson, D., Rohrer, N.: High-performance cmos variability in the 65-nm regime and beyond. IBM Journal of Research and Development 50(4/5), 433–449 (2006)
5. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
6. Chen, Y., Mihcak, M., Kirovski, D.: Certifying authenticity via fiber-infused paper. ACM SIGecom Exchanges 5(3), 29–37 (2005)
7. Corkum, P., Krausz, F.: Attosecond science. Nature Physics 3(6), 381–387 (2007)
8. Dabiri, F., Potkonjak, M.: Hardware aging-based software metering. In: The Design, Automation, and Test in Europe (2009)
9. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22, 644–654 (1976)
10. Friedberg, P., Cao, Y., Cain, J., Wang, R., Rabaey, J., Spanos, C.: Modeling within-die spatial correlation effects for process-design co-optimization. In: Proceedings of the 6th International Symposium on Quality of Electronic Design, pp. 516–521 (2005)
11. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160 (2002)
12. Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, Cambridge (2001)
13. Goulielmakis, E., Yakovlev, V., Cavalieri, A., Uiberacker, V.P.M., Apolonski, A., Kienberger, R., Kleineberg, U., Krausz, F.: Attosecond control and measurement: Lightwave electronics. Science 317, 769–775 (2007)
14. Gustafsson, E., Ruchon, T., Swoboda, M., Remetter, T., Pourtal, E., Lpez-Martens, R., Balcou, P., L'Huillier, A.: Broadband attosecond pulse shaping. Physical Review A 76(1) (2007)
15. In 2008 the number of personal computers will reach billion (2008), `http://www.science.portal.org/in/71` (accessed on February 15, 2009)
16. Kalisz, J.: Review of methods for time interval measurements with picosecond resolution. Metrologia 41(1), 17–32 (2004)
17. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
18. Koushanfar, F., Boufounos, P., Shamsi, D.: Post-silicon timing characterization by compressed sensing. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 185–189 (2008)

19. Koushanfar, F., Qu, G., Potkonjak, M.: Intellectual property metering. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 87–102. Springer, Heidelberg (2001)
20. Lofstrom, K., Daasch, W.R., Taylor, D.: Ic identification circuit using device mismatch. In: IEEE International Solid-State Circuits Conference, pp. 372–373 (2000)
21. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure puf. In: IEEE/ACM International Conference on Computer Aided Design (2008)
22. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: IEEE International Test Conference (2008)
23. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In: IEEE/ACM international conference on Computer-aided design, November 10-14, pp. 721–725 (2002)
24. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
25. Mysyrowicz1, A., Couairon, A., Keller, U.: Self-compression of optical laser pulses by filamentation. New J. Phys. 10, 1–14 (2008)
26. Neureuther, A.: Personal Communication (November 2007)
27. Ozbay, E.: Plasmonics: Merging photonics and electronics at nanoscale dimensions. Science 311, 189–193 (2006)
28. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297(5589), 2026–2030 (2002)
29. Photonic Crystals: Molding the Flow of Light, 2nd edn. Princeton University Press, Princeton (2008)
30. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
31. Roy, S., Asenov, A.: Where do the dopants go? Science 309(5733), 388–390 (2005)
32. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley, Chichester (1996)
33. Shamsi, D., Boufounos, P., Koushanfar, F.: Noninvasive leakage power tomography of integrated circuits by compressive sensing. In: International symposium on Low power electronics and design, pp. 341–346 (2008)
34. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)
35. Thompson, S., Packan, P., Bohr, M.: Mos scaling: Transistor challenges for the 21st century. Intel Technology Journal, Q3, 1–19 (1998)